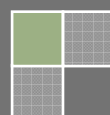
2012

BANCO DE DADOS

Introdução ao estudo de bancos de dados

Este e-book visa elucidar conceitos iniciais para o entendimento dos Bancos de Dados, desde o aspecto conceitual, passando pela modelagem de dados e estruturação de um projeto físico.

Ricardo R. Barcelar

http://www.ricardobarcelar.com.br
ricardobarcelar@email.com.br

APRESENTAÇÃO

O objetivo deste material é nortear o estudo de banco de dados em um curso introdutório, voltado para acadêmicos dos cursos de computação. Os bancos de dados, nesta fase, são estudados tomando-se por base a importância dos Sistemas Gerenciadores de Bancos de Dados (SGBD), os conceitos de Peter Chen acerca da modelagem de dados e da estruturação de um projeto de banco de dados.

ELMASRI e NAVATE são referências no estudo de bancos de dados, e aliado a forma clara e concisa de explicar de HEUSER, este e-book pretende descortinar os conceitos necessários para a estruturação de um banco de dados, partindo do princípio que “sem um projeto adequado de banco de dados não há armazenamento de dados eficiente”.

Ricardo R. Barcelar

Parte
1

HISTÓRIA DOS BANCOS DE DADOS

Em uma primeira abordagem sobre o assunto, é muito importante conhecer como começou toda essa história Bancos de Dados e discutirmos como será o futuro dessa ferramenta tão importante nas empresas e em nossas vidas. Aqui conheceremos os primórdios dos bancos de dados, sua evolução e as perspectivas de futuro.

1.1 DÉCADA DE 60 – PRIMÓRDIOS

Como muitas tecnologias na computação industrial, os fundamentos de bancos de dados surgiram na empresa IBM na década de 1960 através de pesquisas de funções de automação de escritório. Foi um período da história na qual as empresas descobriram que estava muito custoso empregar um número grande de pessoas para fazer trabalhos como armazenar e indexar (organizar) arquivos. Por este motivo, valia a pena os esforços e investimentos em pesquisar um meio mais barato e ter uma solução mecânica eficiente.

Nestes primórdios os dados eram armazenados diretamente em arquivos, os quais implementavam alguns inconvenientes na armazenagem de dados, como:

- redundâncias e inconsistências
- dificuldade de acesso
- falta de integridade lógica
- falta de atomicidade nas transações
- insegurança

Os computadores se tornam parte efetiva do custo das empresas juntamente com o crescimento da capacidade de armazenamento. Foram desenvolvidos dois principais modelos de dados: modelo em rede (CODASYL - Comitee for Data Systems Language) e o modelo hierárquico (IMS – Information Management System). O acesso ao BD é feito através de operações de ponteiros de baixo nível que unem (link) os registros. Detalhes de armazenamento dependiam do tipo de informação a ser armazenada, desta forma, a adição de um campo extra necessitava de uma reescrita dos fundamentos de acesso/modificação do

esquema. Os usuários precisavam conhecer a estrutura física do BD para poder realizar uma consulta.

Um dos primeiros sistemas de banco de dados foi o IMS (*Information Management System*) da IBM lançado no final da década de 60.

1.2 DÉCADA DE 70 – GÊNESIS

Na década de 1970 o pesquisador da IBM - Edgar Frank “Ted” Codd - publicou o primeiro artigo sobre bancos de dados relacionais. Este artigo tratava sobre o uso de cálculo e álgebra relacional para permitir que usuários não técnicos armazenassem e recuperassem uma grande quantidade informações. Codd visionava um sistema onde o usuário seria capaz de acessar as informações através de comandos, nas qual as informações estariam armazenadas em tabelas. Este modelo de dados relacional tornou-se um marco em como pensar em banco de dados. Ele desconectou a estrutura lógica do banco de dados do método de armazenamento físico. Este sistema se tornou padrão desde então.

Devido à natureza técnica deste artigo e a relativa complicação matemática, o significado e proposição do artigo não foram prontamente realizados. Entretanto ele levou a IBM a montar um grupo de pesquisa conhecido como *System R* (Sistema R), que deu origem a um sistema de banco de dados relacional com o mesmo nome.

O Sistema R evoluiu para SQL/DS, que posteriormente passou a chamar o DB2, o mais conhecido sistema gerenciador de banco dos dados da IBM. O Sistema R utilizava uma linguagem chamada *Structured Query Language* (SQL) - Linguagem de Consulta Estruturada. Linguagem esta que se tornou um padrão na indústria para bancos de dados relacionais e hoje em dia é um padrão ISO (*International Organization for Standardization* - Organização Internacional de Padronização.

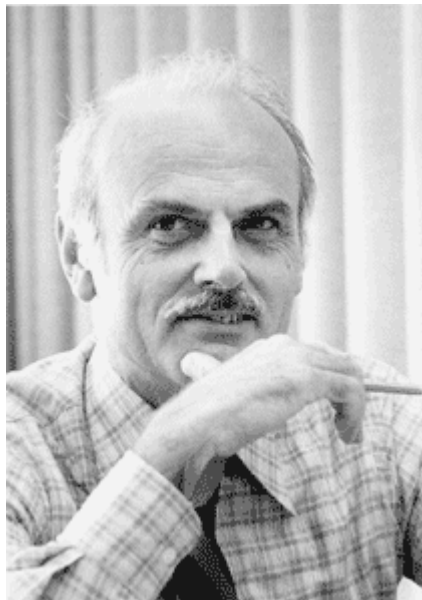



Figura 1 - Dr. Edgar Frank Codd, o pai do modelo relacional.

O Dr. Peter Chen também foi ofereceu grande contribuição ao propor o modelo Entidade-Relacionamento (ER) para projetos de banco de dados dando uma nova e importante percepção dos conceitos de modelos de dados. Assim como as linguagens de alto nível, a modelagem ER possibilita ao projetista concentrar-se apenas na utilização dos dados, sem se preocupar com estrutura lógica de tabelas.

Figura 2 - Dr. Peter Chen, criador do modelo ER

Mesmo a IBM sendo a companhia que inventou o conceito original e o padrão SQL, eles não produziram o primeiro sistema comercial de banco de dados. O feito foi realizado pela *Honeywell Information Systems Inc*., cujo sistema foi lançado em junho de 1976. O sistema era baseado em muitos princípios do sistema que a IBM concebeu, mas foi modelado e implementado fora da IBM.

1.3 DÉCADA DE 80 – DESENVOLVIMENTO

Nesta década surgiu o primeiro sistema de banco de dados construído baseado nos padrões SQL com a empresa Oracle através do Oracle 2 e depois com a IBM através do SQL/DS, servindo como sistema e repositório de informações de outras empresas.

Neste período tornou-se óbvio que existiam várias áreas onde bancos de dados relacionais não eram aplicáveis, por causa dos tipos de dados envolvidos. Estas áreas incluíam medicina, multimídia e física de energia elevada, todas com necessidades de flexibilidade em como os dados seriam representados e acessados. Este fato levou ao início de pesquisas em bancos de dados orientados a objetos, os quais os usuários poderiam definir seus próprios métodos de acesso aos dados e como estes seriam representados e acessados. Ao mesmo tempo, linguagens de programação orientadas a objetos (*Object Oriented Programming* - POO) tais como C++ começaram a surgir na indústria. Isso permitiu com que usuários criassem sistemas de banco de dados para armazenar resultados de pesquisas como o CERN (maior laboratório que trabalha com partículas físicas em pesquisas nucleares europeu) e SLAC (Centro de Aceleração Nuclear Norte-Americano), para mapeamento de rede de provedores de telecomunicações e para armazenar registros médicos de pacientes em hospitais, consultórios e laboratórios.

Foi neste período que os *softwares* de banco de dados relacionais foram sendo refinados graças ao *feedback* (retorno) que os usuários destes sistemas faziam, devido ao

desenvolvimento de sistemas para novas indústrias e ao aumento do uso de computadores pessoais e sistemas distribuídos.

Os pioneiros a implementarem orientação a objeto foram:

- O2 [1988]

- Exodus [1986]

- ORION [1986]

Na Metade da década, os bancos de dados passaram a combinar características de orientação a objeto com o modelo relacional, expandindo a arquitetura com novas possibilidades, como otimização de consultas configuráveis. Por exemplo:

- POSTGRES [1986]

- STARBURST

No final da década há a maturidade da tecnologia, onde os sistemas relacionais passaram a apresentar um desempenho aceitável. São exemplos dessa maturidade:

- DB2

- Ingres

- Oracle

- Sybase

- Informix

Além disso, há a padronização de uma linguagem para manipulação de bancos de dados, o SQL ANSI - *American National Standards Institute* [1986, 1989].

A IBM transforma o DB2 como carro chefe da empresa em produtos para BD. Os modelos em rede e hierárquico passam a ficar em segundo plano praticamente sem desenvolvimentos utilizando seus conceitos, porém vários sistemas legados continuam em uso.

1.4 DÉCADA DE 90 – MATURIDADE

No início da década nota-se a maturidade que os bancos de dados atingiram com o surgimento dos primeiros Sistemas Gerenciadores de Banco de Dados – SGDB Orientados a Objeto comerciais, SGDB's paralelos, tempo real, etc, além de avanços na avanços na padronização de interfaces e interoperabilidade.

O modelo cliente-servidor (*client-server*) passa a ser uma regra para futuras decisões de negócio e vemos o desenvolvimento de ferramentas de produtividade como *Excel/Access* (Microsoft) e ODBC, também é marcado como o início dos protótipos de *Object Database Management Systems* (ODBMS).

Na metade da década surgem novas classes de aplicação:

- *Data Mining*;

- Bibliotecas Digitais (Vídeo sob-demanda, Animação);

- Hipermídia e Multimídia em geral;

- GIS (Sistemas de Informações Geográficas), Meteorologia, Física de Alta Energia (HEP).

Nos idos da década surgem novas abordagens:

- WIIS - *Web Information Integration System*: sistema para tratar dados oriundos de vários *websites*. Lidam com um grande número de w*ebsites*, possui maior autonomia dos componentes e dados semi-estruturados.

- Enfoque de *Data Warehouse* (armazém de dados): dados são extraídos das fontes e armazenados em uma *warehouse(Armazém)*.

- Enfoque de Multi-SGBD: os dados são mantidos nos Web sites, na qual as consultas são decompostas e enviadas aos vários *websites*.

Processos de transação em tempo real (OLTP - *On-Line Transaction Process*) e processos analíticos em tempo real (OLAP – *On-Line Analitical Process*) atingem maturidade através de muitos negócios utilizando os PDVs (Ponto de Venda)

1.5 ONDE ESTAMOS?

Desde sua chegada, os bancos de dados tem tido aumento nos dados de armazenamento, desde os 8 MB (Megabytes) até centenas de petabytes de dados em listas de e-mail, informações sobre consumidores, sobre produtos, vídeos, informações geográficas, etc. Com este aumento de volume de dados, os sistemas de bancos de dados em operação também sofreram aumento em seu tamanho.

Um dos projetos mais ambiciosos de banco de dados já visto está no CERN (Conselho Europeu para Pesquisa Nuclear) que consiste em um banco de dados distribuído com a capacidade de armazenamento na casa dos Hexabytes (1 Hexabyte = 1,000 Petabytes = 1 * 10^{18} Bytes) de dados.

Atualmente, é possível afirmar que a informação gira em torno dos bancos de dados.

O site *Business Intelligence Lowdown* publicou uma matéria em 2007, onde lista os 10 maiores bancos de dados do mundo e nos mostra o porquê desta afirmação:

a) GOOGLE

Ainda não há muito conhecimento sobre a verdadeira dimensão do banco de dados do Google, mas basta considerar que a empresa mantém em *cache* quase todas as páginas da Internet, tem mais de 51 milhões de usuários no Gmail, armazena imagens do mundo todo para o *Google Earth* e para o *Google Maps*, tem milhões de cadastrados no Orkut, entre outros. Isso tudo sem contar que o YouTube faz parte de seu leque de serviços. Não se sabe o quanto de informações o Google armazena, mas estima-se que a empresa ultrapassa tranqüilamente a casa dos petabytes (1 petabyte = 1024 terabytes);

Números: 91 milhões de pesquisas por dia representa 50% de todas as buscas Internet Virtual perfis de inúmeros número de usuários.

b) AT & T

Semelhante a *Sprint*, os Estados Unidos "mais antiga empresa de telecomunicações AT & T mantém um dos maiores bancos de dados. Arquitetonicamente falando, a AT & T a maior base de dados é a "nata" da cultura, uma vez que ostenta títulos, incluindo o maior volume de dados em uma única base (312 terabytes), o segundo maior número de linhas em uma única base (1,9 trilhão), que inclui a AT & T's extensa chamada de registros.

Números: 323 terabytes de informação 1,9 trilhões telefonemas em registros.

c) *World Data Centre for Climate*

Acondicionado em um supercomputador de 35 milhões de euros usado para investigação extensa sobre o clima, capaz de dar a resposta para o aquecimento global. O *World Data Centre for Climate* (WDCC) como o nome indica, o WDCC é uma entidade que faz pesquisas climáticas no mundo todo. Essa é uma das áreas que mais exigem processamento e capacidade de armazenamento de dados. O WDCC possui 220 terabytes de dados facilmente acessíveis na Internet, bem como seus 110 terabytes (ou 24.500 DVD's) usados para simulação de dados, e seis PETABYTES de dados internos da empresa.

Números: 6 PETABYTES é igual à aproximadamente 1.343.296 DVDs.

O que podemos concluir com isso:

- Trivialização do uso da tecnologia de banco de dados?
- Proliferação de produtores e consumidores de dados?
- Aplicações armazenando da ordem de petabytes ou mais?

1.6 E O FUTURO?

O uso de BD móveis são os novos produtos que vem surgindo para comercialização em vários segmentos. Processos de transações distribuídas começam a se tornar uma regra em várias áreas de planejamento de negócios.

Fica a indagação para a quantidade de dados existente no mundo. É sabido que é necessário um novo modelo de armazenamento dados sob pena da escassez de meios para guarda de dados. Sobretudo, vê-se o surgimento de novos modelos de armazenamento de dados que no futuro podem tornar obsoletos os atuais arquétipos de banco de dados.

Seremos além do mais fica a pergunta se seremos capazes de consultar um Banco de Dados de registros médicos/genéticos de um futuro empregado de nossa empresa?

Parte
2

SISTEMAS DE GERÊNCIA DE BANCO DE DADOS

Para entender os Sistemas Gerenciadores de Banco de Dados é importante conhecer alguns conceitos básicos. A primeira definição é relativa aos conceitos de dados e informação. **Dados** são fatos em sua forma primária, os quais podem ser armazenados, como, por exemplo: nome, telefone e endereço. Estes dados ou fatos organizados de maneira significativa e relacionados formam uma **informação**, como por exemplo: os dados das peças em estoque. Assim, é possível obter uma lista de peças em falta.

Sabendo o que são dados e informações é possível definir os **Bancos de Dados** como um conjunto de dados devidamente relacionados capazes de apresentar uma informação.

Os bancos de dados são utilizados em muitas aplicações, abrangendo praticamente todo o campo de programas de computador. Os bancos de dados são mecanismos de armazenamento preferencial para aplicações multiusuário, nas quais é necessário haver coordenação entre vários usuários. Dessa forma, outra definição para banco de dados é que é um conjunto de informações com uma estrutura regular, normalmente, mas não necessariamente, armazenado em algum formato de máquina legível para um computador.

Há uma grande variedade de bancos de dados, desde simples tabelas armazenadas em um único arquivo até um conjunto com milhões de registros armazenados em salas cheias de discos rígidos gerenciados por uma determinada aplicação.

Os dados armazenados em bancos de dados geralmente são apresentados em forma semelhante à uma planilha eletrônica, porém existem sistemas de gestão responsáveis por gerir o armazenamento, classificação, gestão da integridade e recuperação dos dados.

2.1 CARACTERÍSTICAS DE UM BANCO DE DADOS

Como vimos, um banco de dados é uma coleção lógica coerente de dados com um significado inerente. Assim, uma disposição desordenada dos dados não pode ser referenciada como um banco de dados;

Um banco de dados deve ser projetado, construído e populado com dados para um propósito específico;

Deve possuir um conjunto pré-definido de usuários e aplicações;

Representar algum aspecto do mundo real, o qual é chamado de “mini-mundo”. Qualquer alteração efetuada no mini-mundo é automaticamente refletida no banco de dados.

2.2 MODELOS DE BANCO DE DADOS

2.2.1 Modelo Hierárquico ou de árvore

É aquele no qual os dados estão organizados de cima para baixo ou estrutura de árvore invertida. Por exemplo, os dados sobre um projeto para uma empresa podem seguir este tipo de modelo. Este método de ligação é semelhante à relação entre pais e filhos: a criança não existirá sem os pais. É o que mais bem se adapta a situações nas quais as relações lógicas entre os dados podem ser representadas com a abordagem (um-para-muitos).

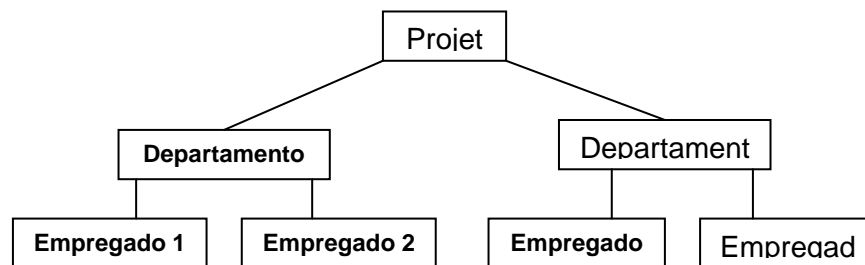



Figura 3 - Modelo Hierárquico

2.2.2 Modelo em Rede

Um modelo em rede é uma extensão do modelo hierárquico. Em vez de se terem apenas vários níveis de relações um-para-muitos, o modelo em rede é uma relação membro-proprietário, na qual um membro pode ter muitos proprietários. Nesse modelo, há freqüentemente mais de um caminho pelo qual um determinado elemento de dado pode ser acessado.

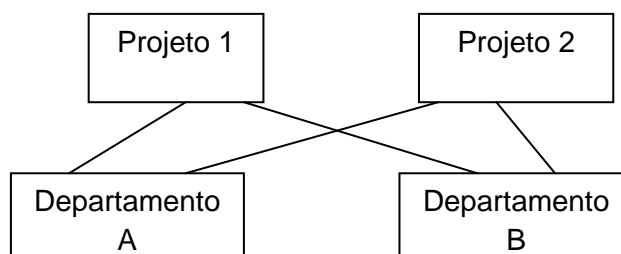



Figura 4 - Modelo em Rede

2.2.3 Modelo Relacional

Os modelos relacionais se tornaram os mais populares. A finalidade global deste modelo é descrever o dado usando um formato tabular padrão (todos os elementos são localizados em tabelas bidimensionais). As tabelas organizam os dados em linhas e colunas, simplificando o acesso e a manipulação dos dados.

Uma vez colocados os dados no Banco de Dados Relacional, pode-se fazer perguntas e manipular dados utilizando as operações da álgebra relacional. As manipulações básicas de dados incluem a sua seleção, projeção e agrupamento. Outros aspectos deste modelo serão abordados nos capítulos seguintes.

2.3 SISTEMA DE GERENCIADOR DE BANCO DE DADOS (SGBD)

Um banco de dados pode ser criado e mantido por um conjunto de aplicações desenvolvidas especialmente para esta tarefa ou por um Sistema Gerenciador de Banco de Dados. Este **é um software com recursos específicos para facilitar a manipulação das informações dos bancos de dados e o desenvolvimento de programas aplicativos**.

Outro conceito interessante sobre SGBD pode ser encontrado no Wikipédia:

> *...o conjunto de programas de computador (softwares) responsáveis pelo gerenciamento de uma base de dados.*

Os Sistemas Gerenciadores de Bancos de Dados surgiram para atender à necessidade de armazenamento e de recuperação de grandes volumes de informações, propiciando um ambiente seguro e adequado.

Seu **principal objetivo** é retirar da aplicação cliente a responsabilidade de gerenciar o acesso, manipulação e organização dos dados. Estes sistemas disponibilizam uma interface para que os seus clientes possam incluir, alterar ou consultar dados.

2.3.1 Funções Básicas de um Sistema Gerenciador de Banco de Dados

Como todo e qualquer aplicativo os sistemas gerenciadores de bancos de dados dispõe de funções que são compartilhadas por qualquer tipo de sistema gerenciador de banco de dados, independente de seu fabricante. Dentre elas, destacam-se:

- Proporcionar maior abstração, isolando o usuário dos pormenores internos de como os dados estão armazenados;

- Rapidez no acesso às informações;

- Proporcionar independência dos dados em relação aos programas de recuperação, cuja estrutura física de armazenamento independe da estratégia de acesso;

- Compartilhar a base de dados entre vários aplicativos, em que diferentes tipos de interfaces atendem às necessidades de distintos usuários;

- Maior facilidade de realizar cópias de segurança;
- Proporcionar a comunicação diretamente com um software ou servidor;
- Proporcionar integridade dos dados.

2.3.2 Arquitetura de um Sistema Gerenciador de Banco de Dados

Atualmente, existem várias tendências para arquitetura de Banco de Dados. A arquitetura mais conhecida é a ANSI/SPARC. Esta arquitetura apresenta-se fundamentada em três níveis onde cada um desses níveis corresponde às abstrações dos dados armazenados no banco de dados.

Os esquemas são definidos como:

a) Nível externo: ou esquema de visão, o qual descreve o modo pelo qual os dados são vistos pelos usuários do sistema gerenciador de banco de dados. Cada visão descreve quais porções do banco de dados um usuário ou grupo de usuários terá acesso.

b) Nível conceitual: ou esquema conceitual, o qual descreve a estrutura do banco de dados como um todo, inclusive como os dados se relacionam. É uma descrição global do banco de dados que não fornece detalhes do modo como os dados estão fisicamente armazenados.

c) Nível interno ou físico: ou esquema interno, o qual descreve a estrutura de armazenamento físico do banco de dados. Utiliza um modelo de dados e descreve detalhadamente os dados armazenados e os caminhos de acesso ao banco de dados. É a descrição mais próxima de como os dados serão armazenados.

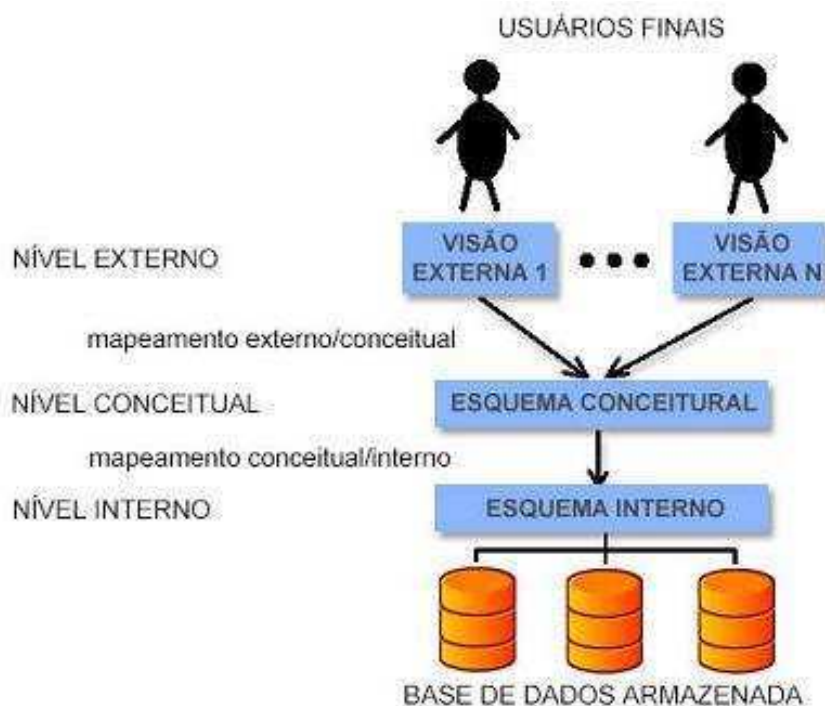



Figura 5 - Arquitetura de SGBD

Partindo da arquitetura proposta é importante conhecer alguns conceitos diretamente relacionados:

a) Abstração de Dados: Um Sistema Gerenciador de Banco de Dados é composto de uma coleção de arquivos inter-relacionados e de um conjunto de programas que permitem aos usuários realizarem acesso aos arquivos e modificá-los. O grande objetivo de um sistema de banco de dados é prover aos usuários uma **visão abstrata dos dados**. Isto é, **o sistema omite certos detalhes de como os dados são armazenados e mantidos**. No entanto, para que o sistema possa ser utilizado, os dados devem ser buscados de forma eficiente. Este conceito tem direcionado o projeto de estrutura de dados complexas para a representação de dados em um banco de dados. Uma vez que muitos dos usuários de banco de dados não são treinados para computação, a complexidade está escondida desses usuários através de **diversos níveis de abstração** que simplificam a interação do usuário com o sistema.

b) Nível de Visões: O mais alto nível de abstração descreve apenas parte do banco de dados. Apesar do uso de estruturas mais simples do que no nível conceitual, alguma complexidade perdura devido ao grande tamanho do banco de dados. Muitos usuários do sistema de banco de dados não estarão interessados em todas as informações. Em vez disso precisam de apenas uma parte do banco de dados. O nível de abstração das visões de dados é definido para simplificar esta interação com o sistema, que pode fornecer muitas visões para o mesmo banco de dados.

Dentro deste conceito, no **nível físico** um registro *cliente*, *conta* ou *funcionário* pode ser descrito como um bloco de posições de armazenamento consecutivo (por exemplo, palavras ou bytes). No **nível conceitual**, cada registro destes é descrito por uma definição de tipo e pelo inter-relacionamento entre esses tipos de registros. No **nível de visões**, diversas visões do banco de dados são definidas, por exemplo: os contadores de um banco vêem apenas a parte do banco de dados que possui informações sobre contas dos clientes. Eles não podem ter acesso a informações que se referem a salário dos funcionários.

c) Independência de Dados: Conhecemos três níveis de abstração pelos quais o banco de dados pode ser visto. A habilidade de modificar a definição de um esquema em um nível sem afetar a definição de esquema num nível mais alto é chamada de *independência de dados*.

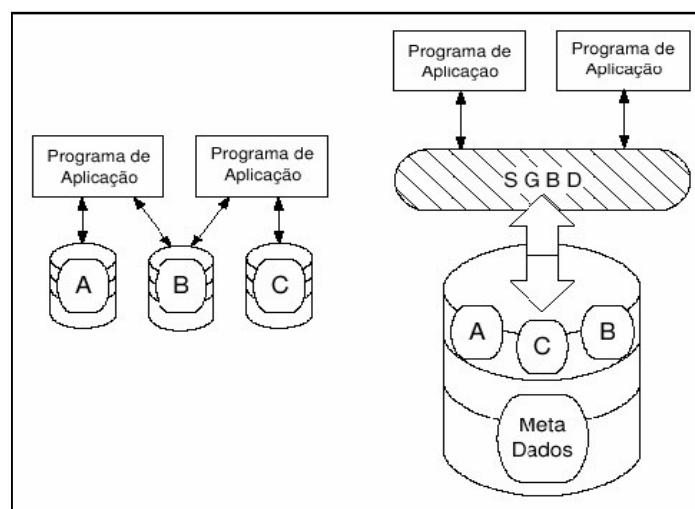



Figura 6 – Independência de dados

Existem dois níveis de independência dos dados:

- **Independência Física de Dados:** É a habilidade de modificar o esquema físico sem a necessidade de reescrever os programas aplicativos. As modificações no nível físico são ocasionalmente necessárias para melhorar o desempenho;

- **Independência Lógica de Dados:** É a habilidade de modificar o esquema conceitual sem a necessidade de reescrever os programas aplicativos. As modificações no nível conceitual são necessárias quando a estrutura lógica do banco de dados é alterada (por exemplo, a adição de contas de bolsas de mercado num sistema bancário).

A independência lógica dos dados é mais difícil de ser alcançada do que a independência física, porém os programas são bastante dependentes da estrutura lógica dos dados que eles acessam.

Este conceito de independência dos dados é similar em muitos aspectos ao conceito de tipos abstratos de dados em modernas linguagens de programação. Ambos escondem detalhes de implementação do usuário. Isto permite ao usuário concentrar-se na estrutura geral em vez de detalhes de baixo nível de implementação.

d) Dicionário de Dados: Um dicionário de dados é uma **coleção de metadados** que contêm definições e representações de elementos de dados. Dentro do contexto de sistemas gerenciadores de bancos de dados, um dicionário de dados é um grupo de tabelas, habilitadas apenas para leitura ou consulta, ou seja, é uma base de dados, propriamente dita, que entre outras coisas, mantém as seguintes informações:

- Definição precisa sobre elementos de dados
- Perfis de usuários, papéis e privilégios
- Descrição de objetos
- Integridade de restrições
- *Stored procedures*[1] e gatilhos
- Estrutura geral da base de dados
- Informação de verificação
- Alocações de espaço

Um dos benefícios de um dicionário de dados bem preparado é a consistência entre itens de dados através de diferentes tabelas. Por exemplo, diversas tabelas podem conter números de telefones. Utilizando uma definição de um dicionário de dados bem feito, o formato do campo 'número de telefone' definido com "(99)9999-9999" deverá ser obedecido em todas as tabelas que utilizarem esta informação.

---

[1] pequeno trecho de programa de computador, armazenado em um SGBD, que pode ser chamado freqüentemente por um programa principal.

Os dicionários de dados são gerados, normalmente, separados do Modelo de Dados visto que estes últimos costumam incluir complexos relacionamentos entre elementos de dados.

e) Modelo de Dados: Uma das principais características da abordagem de banco de dados é que fornece alguns níveis de abstração de dados omitindo ao usuário final detalhes de como estes dados são armazenados. Um modelo de dados é um conjunto de conceitos que podem ser utilizados para descrever a estrutura lógica e física de um banco de dados. Por estrutura podemos compreender o tipo dos dados, os relacionamentos e as restrições que podem recair sobre os dados.

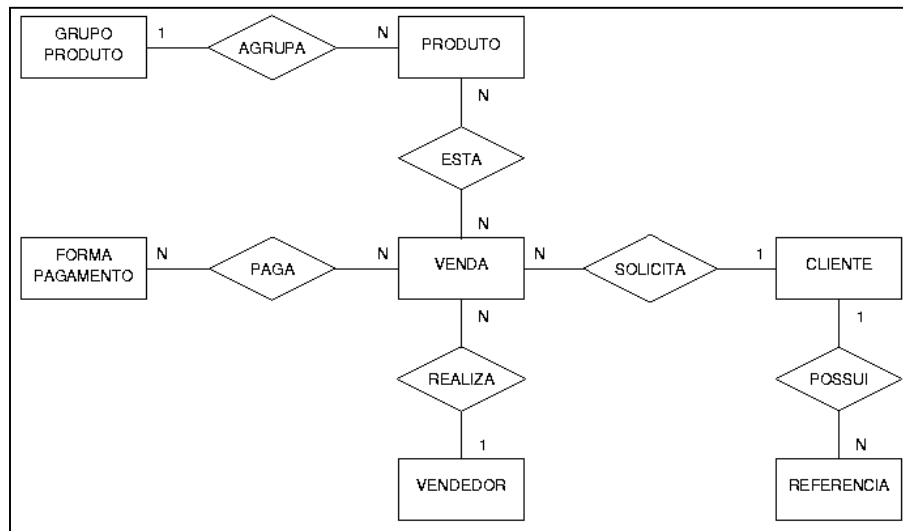



Figura 7 - Modelo Entidade-Relacionamento

Os modelos de dados podem ser basicamente de dois tipos:

- **Alto nível:** ou modelo de dados conceitual, que fornece uma visão mais próxima do modo como os usuários visualizam os dados realmente;

- **Baixo nível:** ou modelo de dados lógico ou físico, que fornece uma visão mais detalhada do modo como os dados estão realmente armazenados no computador.

f) Esquemas e Instâncias: Em qualquer modelo de dados utilizado, é importante distinguir a descrição do banco de dados do banco de dados por si próprio. A descrição de um banco de dados é chamada de **esquema de um banco de dados** e é especificada durante o projeto do banco de dados. Geralmente, poucas mudanças ocorrem no esquema do banco de dados. Os dados armazenados em um determinado instante do tempo formam um conjunto chamado de **instância do banco de dados**. A instância altera toda vez que uma alteração no banco de dados é feita.

O Sistema Gerenciador de Banco de Dados é responsável por garantir que toda instância do banco de dados satisfaça ao esquema do banco de dados, respeitando sua estrutura e suas restrições. O esquema de um banco de dados também pode ser chamado de **intensão** de um banco de dados e a instância de **extensão** de um banco de dados.

g) As Linguagens para Manipulação de Dados: Para a definição dos esquemas conceitual e interno pode-se utilizar uma linguagem chamada DDL (*Data Definition Language -*

Linguagem de Definição de Dados). O Sistema Gerenciador de Banco de Dados possui um compilador DDL que permite a execução das declarações para identificar as descrições dos esquemas e para armazená-las no catálogo do Sistema Gerenciador. A DDL é utilizada onde a separação entre os níveis interno e conceitual não é muito clara. Onde a separação entre os níveis conceitual e interno são bem claras, é utilizada outra linguagem, a SDL (*Storage Definition Language* - Linguagem de Definição de Armazenamento) para a especificação do esquema interno. A especificação do esquema conceitual fica por conta da DDL.

Em Sistemas Gerenciadores de Bancos de Dados que utilizam a arquitetura de três esquemas ou nível, é necessária a utilização de mais uma linguagem para a definição de visões, a VDL (*Vision Definition Language* - Linguagem de Definição de Visões).

Uma vez que o esquema esteja compilado e o banco de dados esteja populado, utiliza-se uma linguagem para fazer a manipulação dos dados, a DML (*Data Manipulation Language* - Linguagem de Manipulação de Dados).

2.3.3 Vantagens do uso de um Sistema Gerenciador de Banco de Dados

A utilização de Sistemas de Bancos de Dados tornou-se um padrão desde quando na década de 60 percebeu-se a inviabilidade de gerenciar um volume muito grande de informação sem um sistema de computador. A utilização de Sistemas Gerenciadores de Bancos de Dados trouxe várias vantagens, dentre elas:

a) Controle de Redundância: No processamento tradicional de arquivos, cada grupo de usuários deve manter seu próprio conjunto de arquivos e dados. Desta forma, acaba ocorrendo redundâncias que prejudicam o sistema com problemas como:

- Toda vez que for necessário atualizar um arquivo de um grupo, então todos os grupos devem ser atualizados para manter a integridade dos dados no ambiente como um todo;

- A redundância desnecessária de dados leva ao armazenamento excessivo de informações, ocupando espaço que poderia ser utilizado com outras informações.

b) Compartilhamento de Dados: Um Sistema Gerenciador de Banco de Dados multiusuário deve permitir que usuários distintos acessem o banco de dados ao mesmo tempo. Este fator é essencial para que múltiplas aplicações integradas possam manipular seus dados. Dessa forma é necessário manter o controle da concorrência para assegurar que o resultado de atualizações sejam corretos. Um banco de dados multiusuários deve fornecer recursos para a construção de múltiplas visões.

c) Restrição a Acesso não Autorizado: Um Sistema Gerenciador de Banco de Dados deve fornecer um subsistema de autorização e segurança, o qual é utilizado pelo DBA para criar contas e especificar as restrições destas contas. O controle de restrições se aplica tanto ao acesso aos dados quanto ao uso de *softwares* inerentes ao Sistema Gerenciador de Banco de Dados.

d) Representação de Relacionamentos Complexos entre Dados: Um banco de dados pode incluir uma variedade de dados que estão interrelacionados de várias formas. O Sistema Gerenciador de Banco de Dados deve fornecer recursos para se representar uma grande variedade de relacionamentos entre os dados, bem como recuperar e atualizar os dados de maneira prática e eficiente.

e) Tolerância a Falhas: Um Sistema Gerenciador de Banco de Dados deve fornecer recursos para recuperação de falhas tanto de software quanto de hardware.

Por fim e para melhor compreensão é interessante analisar e compreender a estrutura de um Sistema Gerenciador de Banco de Dados conforme figura 8 e 9.

Esta figura, de forma simplificada, mostra os componentes típicos de um SGBD. O banco de dados e o catálogo de dados são normalmente armazenados no disco. O acesso ao disco é controlado principalmente pelo sistema operacional, que organiza as entradas e as saídas. Um módulo de gerenciamento dos dados armazenados de alto nível do SGBD controla o acesso à informação do SGBD que está armazenada no disco, se for parte do banco de dados ou do catálogo.

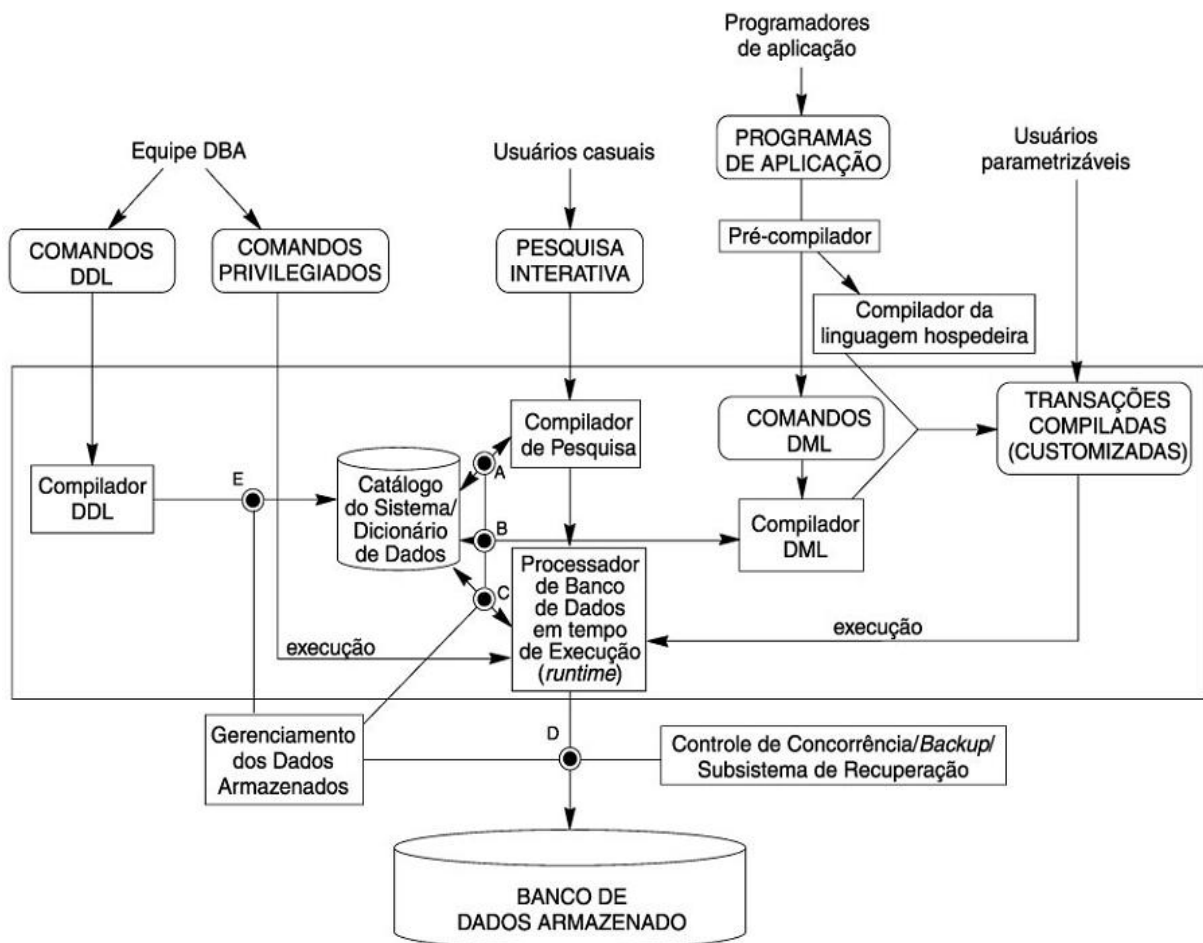



Figura 8 – Módulos de um SGBD

A figura 9, logo abaixo apresenta outra visão, desta vez da arquitetura interna de um SGBD centralizado.

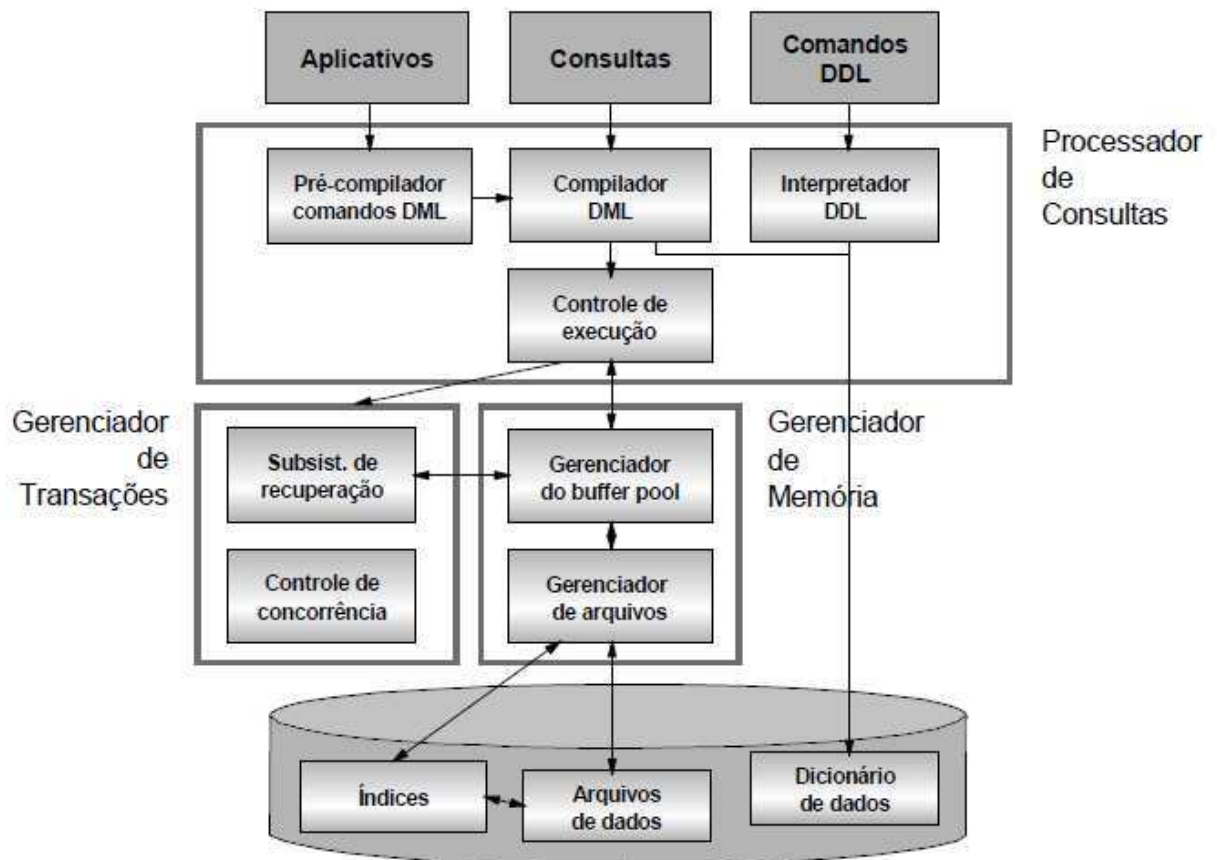



Figura 9 - Arquitetura de SGBD

Parte
3

MODELAGEM DE DADOS

Uma das principais características da abordagem de banco de dados é que ela fornece alguns níveis de abstração de dados omitindo ao usuário final detalhes de como os dados são armazenados.

Define-se como modelo de dados um conjunto de conceitos que podem ser utilizados para descrever a estrutura lógica e física de um banco de dados.

3.1 ETAPAS DA MODELAGEM DE DADOS

Três são as etapas da modelagem de banco de dados:

- Projeto Conceitual

- Projeto Lógico

- Projeto Físico

Contudo, uma etapa não descrita, mas de suma importância para qualquer etapa da modelagem de dados é a **análise de requisitos** que representa uma etapa onde serão coletadas as informações de uma abstração do mundo real – o minimundo.

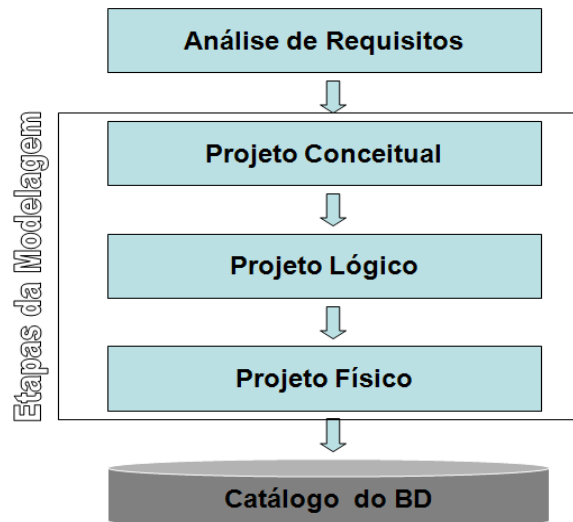



Figura 10 - Etapas da modelagem de dados

3.1.1 Projeto Conceitual

É a descrição de mais alto nível da estrutura do BD, não contendo detalhes de implementação; Nesta etapa não é necessário se preocupar com o tipo de SGBD a ser usado, ou seja o projeto é independente do tipo de SGBD usado;

É o ponto de partida do projeto de Banco de Dados e seu objetivo é representar a semântica da informação, independente de considerações de eficiência.

O objetivo é a representação dos requisitos de dados do domínio.

Requisitos: Clareza (facilidade de compreensão) e exatidão (formal).

3.1.2 Projeto Lógico

No modelo lógico existe a descrição da estrutura do BD que pode ser processada pelo SGBD. Em poucas palavras é o modelo conceitual mapeado para um modelo lógico de dados; Nesta etapa há a dependência da classe de modelos de dados utilizada pelo SGBD, mas não do SGBD.

A ênfase do modelo lógico está na eficiência de armazenamento, ou seja, em evitar muitas tabelas (e junções); tabelas subutilizadas, etc.

Futuras alterações no modelo lógico devem ser primeiro efetuadas no Modelo Conceitual.

3.1.3. Projeto Físico

Nesta etapa ocorre o mapeamento do modelo lógico em um esquema físico de acordo com o SGBD específico, ou seja, o modelo criado está diretamente ligado ao SGBD escolhido. No modelo físico contém a descrição da implementação da base de dados na qual descreve as estruturas de armazenamento e os métodos de acesso. Caracteriza-se pela criação do

esquema SQL da modelagem lógica. Sua ênfase na eficiência de acesso como na implementação de consultas, índices, etc.

Exemplos: alocação dinâmica de espaços, clusterização, particionamento físico das tabelas, etc.

3.2 ABORDAGEM ENTIDADE-RELACIONAMENTO (ER)

A abordagem entidade-relacionamento é um padrão para a modelagem conceitual. Foi criada em 1976 por Peter Chen que junto com alguns conceitos apresenta uma **notação gráfica para diagramas** que tem por características:

- Ser um modelo simples, com poucos conceitos;

- Representação gráfica de fácil compreensão.

Um esquema conceitual de dados é também chamado de esquema ER, diagrama ER, ou modelo ER.

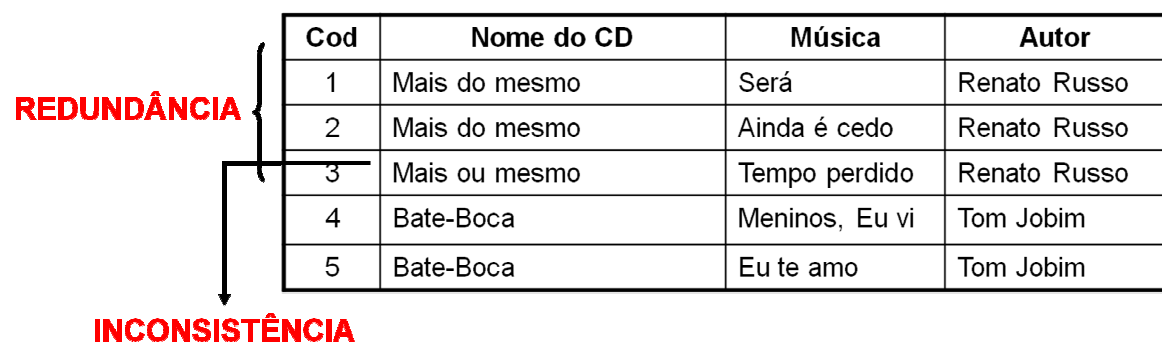
É um modelo conceitual que representa os elementos do domínio do problema e, consequentemente, não considera questões tecnológicas. Assim, alguns dos elementos descritos neste modelo não possuem correspondência com os recursos oferecidos pelos bancos de dados relacionais, tornando necessário transformar o Modelo Entidade-Relacionamento em uma notação que possa ser implementada neste tipo de banco de dados.

3.2.1 Abordagem Relacional

A abordagem relacional é a utilização de conceitos de entidade e relacionamento para criar as estruturas que irão compor o BD. Partindo da necessidade do usuário ou grupo de usuários do sistema, iniciamos a pesquisa das necessidades de informação desses usuários, o que chamamos de levantamento de requisitos. **A definição do escopo do sistema é importante para o início do trabalho de análise de dados**.

É comum no início do desenvolvimento de um sistema não termos a noção exata da tarefa a ser realizada. **O maior erro nesta fase é admitir que já sabemos o que deve ser feito.**

Para minimizar esse problema, devemos criar uma estrutura gráfica que permita identificar as entidades de um sistema e como estas se relacionam.

O modelo de dados dará suporte a empresa, incorporando as informações necessárias para o andamento dos negócios. Ele será composto, basicamente, de Entidades e Relacionamentos daí ser conhecido como **Modelo Entidade-Relacionamento (MER)**.

3.2.2 Vantagens na utilização do MER

- Sintaxe Robusta: o modelo documenta as necessidades de informação da empresa de maneira precisa e clara.

- Comunicação com o usuário: os usuários podem, com pouco esforço, entender o modelo.
- Facilidade de criação: pode-se criar e manter o modelo com facilidade.
- Integração com várias aplicações: diversos projetos podem ser inter-relacionados.
- Utilização universal: o modelo não está vinculado a um BD, garantindo independência de implementação.

3.2.3 Objetivo da Modelagem de dados

Desenvolver um modelo que, contendo entidades e relacionamentos, seja capaz de representar os requerimentos das informações do negócio, evitando redundâncias, inconsistências e economia de espaço.

Cod	Nome do CD	Música	Autor
1	Mais do mesmo	Será	Renato Russo
2	Mais do mesmo	Ainda é cedo	Renato Russo
3	Mais ou mesmo	Tempo perdido	Renato Russo
4	Bate-Boca	Meninos, Eu vi	Tom Jobim
5	Bate-Boca	Eu te amo	Tom Jobim

Figura 11 - Representação de dados

3.2.4 Objetos Conceituais

A Abordagem Entidade-Relacionamento (ER) é a técnica mais utilizada e difundida que existe.

O modelo de dados é representado através de um modelo entidade-relacionamento (MER), que graficamente é chamado de Diagrama entidade-relacionamento (DER). Chen destaca a importância de reconhecer objetos do negócio e os classificou em dois grupos: **Entidades e Relacionamentos**.

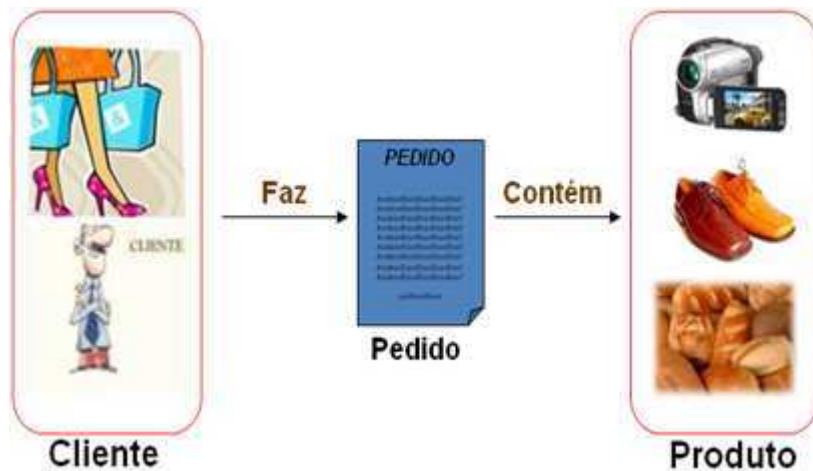



Figura 12 – Entidade/Relacionamento

3.2.4.1 Entidades

Entidades são objetos que existem no mundo real com uma identificação distinta e com um significado próprio. Também são descritas como objetos da realidade na qual se deseja manter informações no banco de dados. Normalmente é representado por um **substantivo** na descrição do negócio.

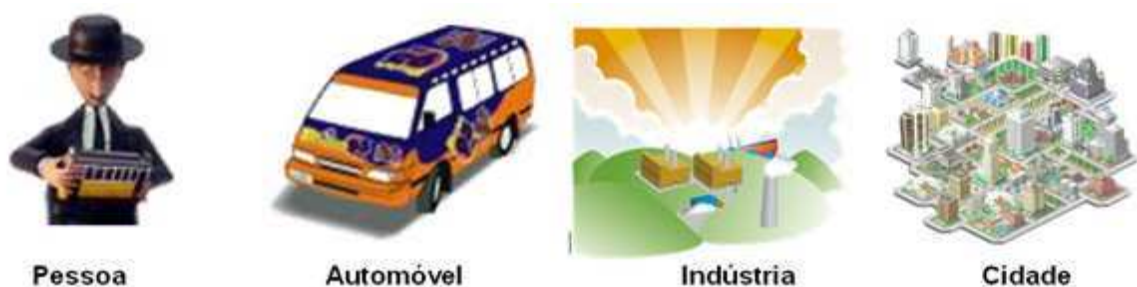



Figura 13 - Exemplos de entidades

Em outras palavras são as **coisas** que existem no negócio.

É importante ressaltar que uma entidade não é caracterizada somente por objetos físicos, podendo existir objetos abstratos neste conceito. Observe esta pequena estória:

O Sr. Joaquim sente fortes dores no peito e procura um consultório médico para se consultar. Chegando ao consultório ele se apresenta e a secretária faz um pequeno cadastro com seus dados e sem seguida o encaminha para ser atendido por um médico. Depois de realizada a consulta, o médico receita-lhe alguns medicamentos.

Pergunta: Qual objeto abstrato é possível armazenar alguma informação?

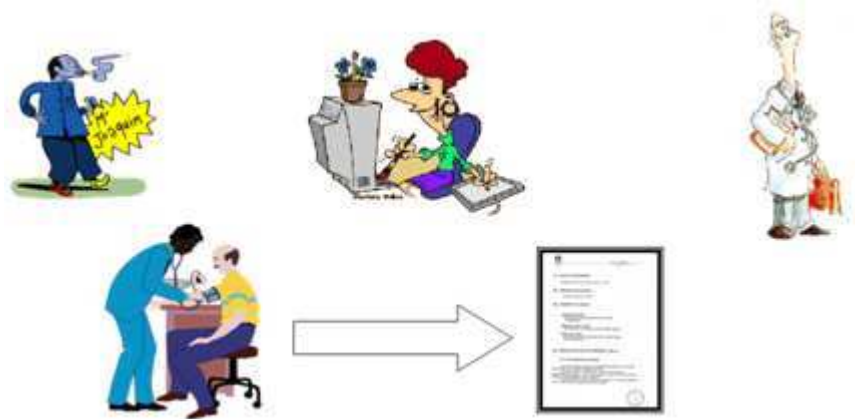



Figura 14 - Entidades

Analisando o minimundo descrito acima é possível identificar objetos abstratos e concretos: Médico e paciente são caracterizados como objetos concretos, mais fáceis de serem identificados. Um fato que se deseja registrar que possua características próprias como a **consulta médica** são caracterizados como objetos abstratos.

NOTAÇÃO:

Em um Diagrama Entidade-Relacionamento uma entidade é representada através de **retângulo** contendo o nome da entidade, como no exemplo abaixo:

Figura 15 - Notação de entidade

3.2.4.2 Atributos

São informações que qualificam uma entidade e descrevem seus elementos ou características. Quanto transpostos para o modelo físico são chamados de colunas ou campos.

Um atributo é uma característica, logo não contém um grupo de informações.

É importante utilizar sempre uma visão espacial de dados, a fim de enxergar o todo e não uma única ocorrência. Existem diversos tipos de atributo, dentre eles:

- Atributos simples
- Compostos
- Multivalorados
- Especiais

Os **atributos compostos** podem ser divididos em subpartes menores que representam outros atributos básicos com significados diferentes. Por exemplo, o atributo Endereço, que pode ser subdividido em número, logradouro, cidade, estado e CEP. Os atributos que não são divisíveis são chamados **atributos simples**.

A maioria dos atributos possui um único valor. Em alguns casos, um atributo pode ter um conjunto de valores para a mesma entidade, como por exemplo, o atributo cores ou o atributo formação. Esses atributos são chamados de **multivalorados**.

MÉDICO

CRM_Médico	Nome_medico	Especialidade_medico
212121	Luís Junior	Pediatra
323232	Pascoal Ramos	Neurologista
434343	Patrícia Silva	Cardiologista

Figura 16 - Atributo

Normalmente existem atributos que tem funções **especiais** em uma entidade. Dessas algumas servem como identificadores, a saber:

- Chave primária: É o atributo ou grupamento de atributos cujo valor identifica unicamente uma entidade dentre todas as outras. Deve ter conteúdo reduzido e valor constante no tempo. Pode ser natural ou artificial.

- **Chave candidata:** É o atributo ou grupamento de atributos que tem a propriedade de identificação única. Pode vir a ser a chave primária.

- **Chave estrangeira:** É quando um atributo de uma entidade é a chave primária de outra entidade com a qual ela se relaciona.

- **Chave composta:** É formada pelo grupamento de mais de um atributo.

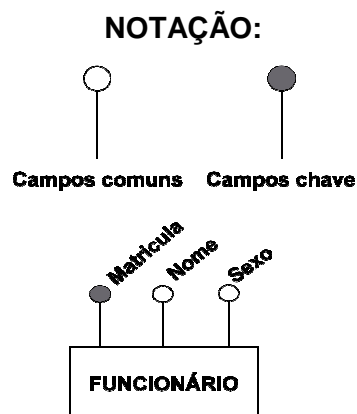



Figura 17 - Notação de atributo

Ainda em relação ao atributo, este pode ser classificado como **multivalorado** quando, este tipo possui mais de um valor para cada atributo a partir dele próprio. Um tipo de atributo multivalorado pode ser organizado na prática como uma lista, conjunto ou coleção de elementos.

3.2.4.3 Tuplas

Os atributos e seu valores descrevem as instâncias de uma entidade, formando o que chamamos de tuplas ou registros.

CONSULTA

Data_consulta	CRM_Medico	Paciente
01/02/2005	212121	Joaquim Rodrigues
04/12/2005	323232	Rodrigo Tristão
05/02/2006	434343	Evandro Marca Passos

Registro

Figura 18 - Tuplas

Não devemos considerar como entidade um objeto, se não conseguirmos ter a visão de seu conteúdo em instâncias com valores de atributos – Tuplas.

3.2.4.4 Relacionamentos

É o fato ou acontecimento que liga dois objetos existentes no mundo real, ou seja, o fato que efetua a junção de duas ou mais tabelas.

NOTAÇÃO:

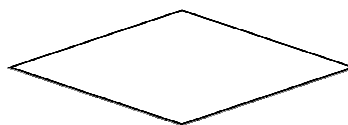



Figura 19 - Notação de relacionamento

Várias são as possibilidades de relacionamentos, como serão vistos a frente. Um relacionamento é caracterizado por um **verbo**, como: Pessoas **moram** em Apartamentos.

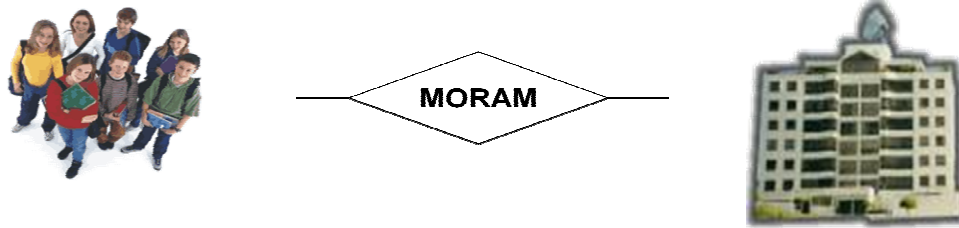



Figura 20 - Exemplo de relacionamento

3.2.4.5 Classificação dos Relacionamentos

a) Quanto a Cardinalidade ou grau dos relacionamentos:

- 1:1 (Um para um)
- 1:N (Um para muitos)
- N:N (muitos para muitos)

- Relacionamento um-para-um: cada elemento de uma entidade relaciona-se com um e somente um elemento de outra entidade.

Figura 21 - Relacionamento 1:1

- Relacionamento um-para-muitos: cada elemento de uma entidade relaciona-se com muitos elementos de outra entidade. É o mais comum no mundo real.

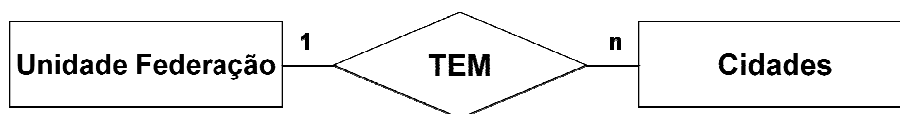



Figura 22 - Relacionamento 1:N

- Relacionamento muitos-para-muitos: caracteriza-se pelo relacionamento possuir dados que são inerentes ao fato e não às entidades.

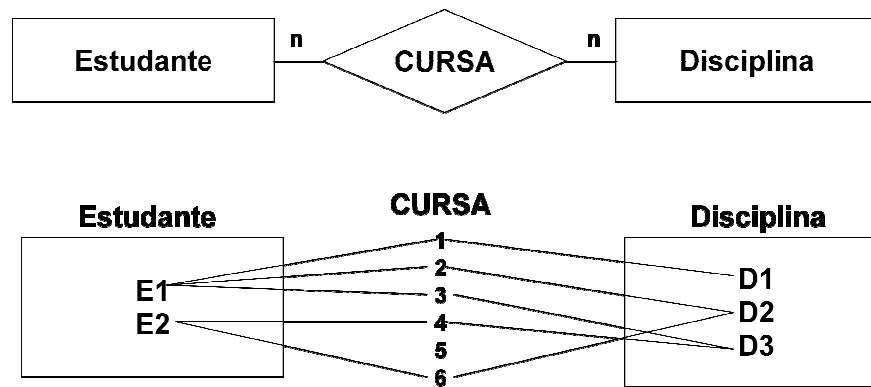



Figura 23 - Relacionamento N:N

Em suma, na figura 18 estão representados os tipos de relacionamentos com sua representação baseada na teoria dos conjuntos.

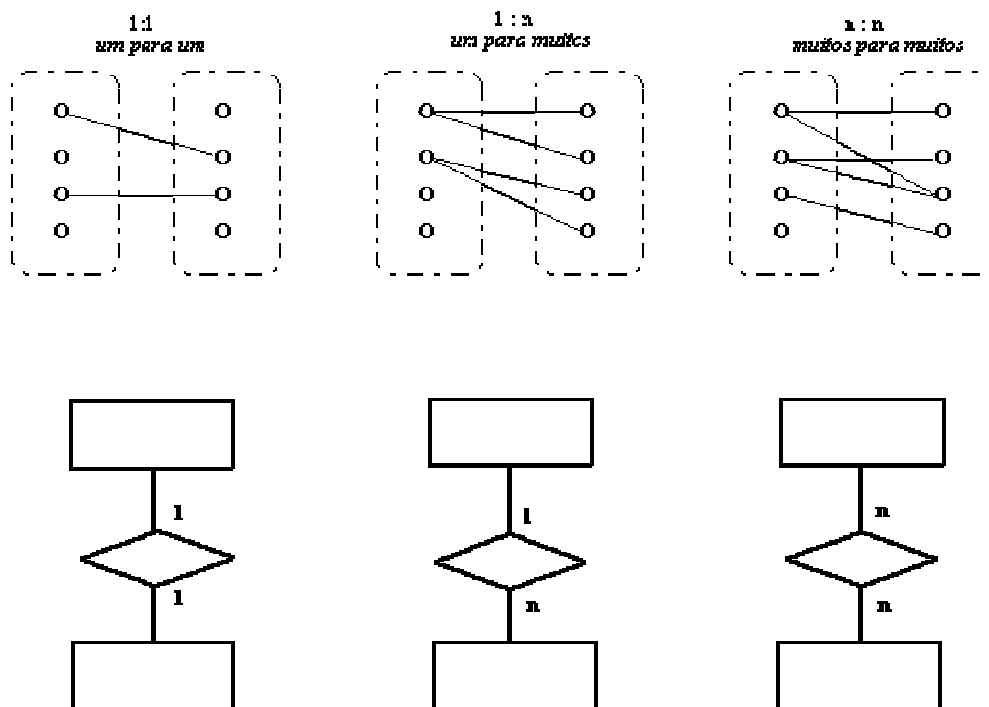



Figura 24 - Relacionamentos quanto a cardinalidade

Todos os relacionamentos vistos até agora se referem a **relacionamentos binários**, ou seja, representam relacionamentos entre duas entidades, conforme figura 19.

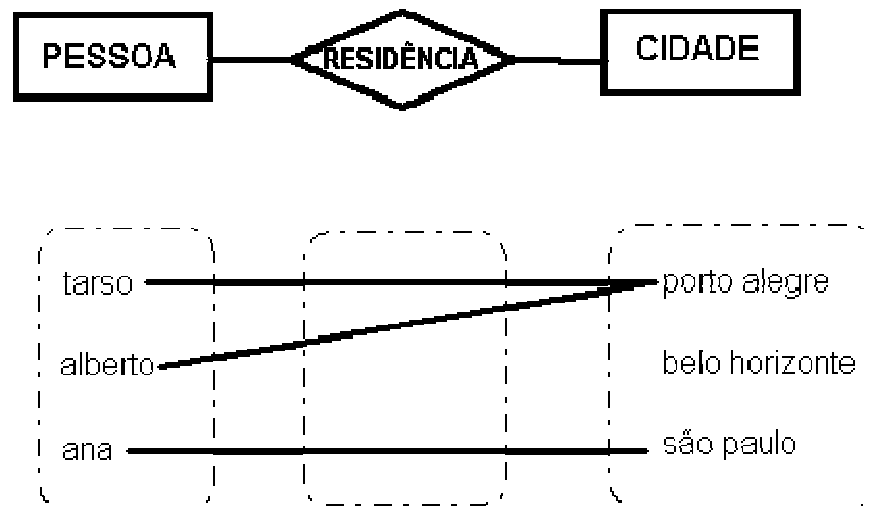



Figura 25 - Relacionamento binário

A cardinalidade pode também ser representada de acordo com a cardinalidade máxima ou mínima conforme figuras 20 e 21.

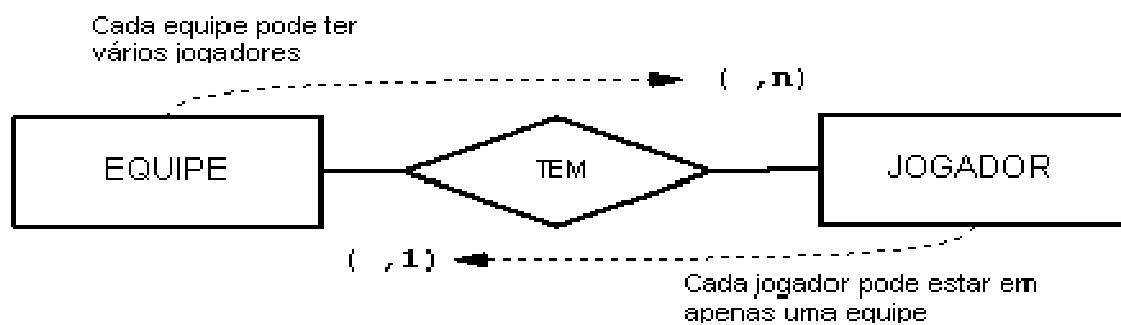



Figura 26 - Cardinalidade Máxima

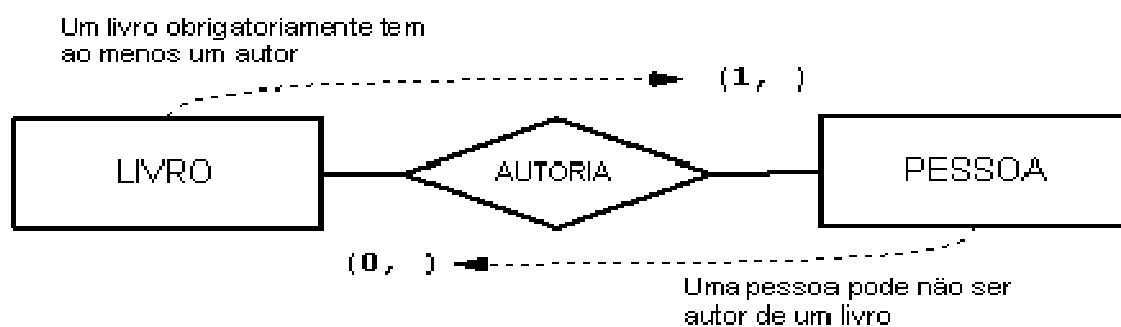



Figura 27 - Cardinalidade Mínima

b) Quanto a natureza

Indica se as ocorrências de uma entidade participam de forma Opcional ou Compulsória.

- Compulsória
- Opcional

NOTAÇÃO:

Figura 28 - Notação de relacionamento quanto a natureza

Figura 29 - Exemplo de classificação quanto a natureza

3.4.4.6. Auto-Relacionamento

Cada elemento de uma entidade relaciona-se com um ou mais elementos da mesma entidade, ou seja, demonstra o relacionamento de ocorrências de uma entidade com outras ocorrências da mesma entidade. São definidos papeis de cada lado do relacionamento.

Exemplos: Peça, Pessoa, Funcionário, etc.

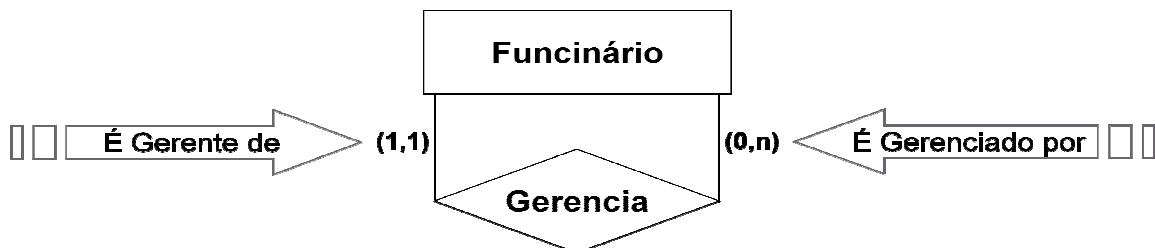



Figura 30 - Auto-relacionamento

3.4.4.7. Relacionamento Ternário

Embora a maioria dos relacionamentos ocorra entre duas entidades (relacionamentos binários) podem ser definidos relacionamentos entre qualquer número de entidades.

Funcionário	Projeto	Função
Ana	Anaconda	Análise de Sistemas
Ana	Sucuri	Programação
Carlos	Sucuri	Análise de Sistemas
Carlos	Sucuri	Programação

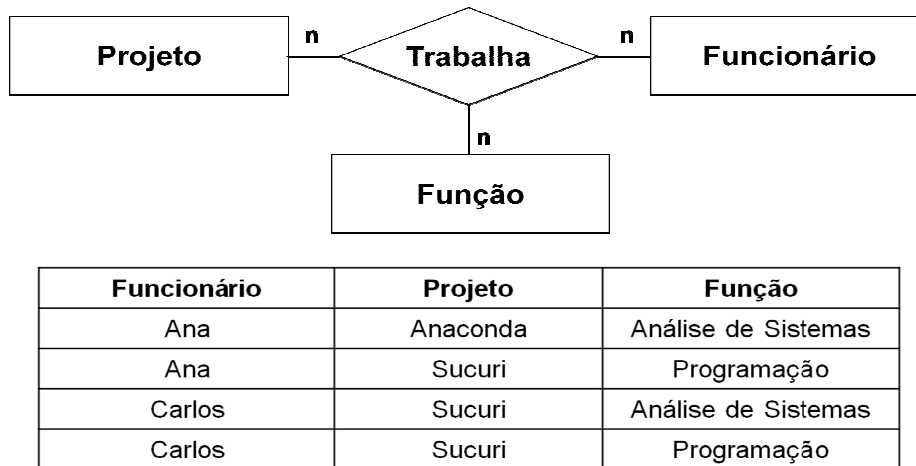



Figura 31 - Relacionamento ternário

Os Relacionamentos ternários devem ser utilizados com **muito cuidado**, pois muitas vezes induzem a criação de bancos de dados **não normalizados**. Como regra geral deve-se criar um relacionamento ternário apenas quando **não for possível representar a regra de negócio** desejada em um ou mais relacionamentos binários.

3.2.4.8. Entidade Forte e Entidade Fraca

É possível que um conjunto de entidades não tenha atributos suficientes para formar uma chave primária. Tal conjunto de entidades é nomeado como conjunto de **entidades fracas**. Um conjunto de entidades que possui uma chave primária é definido como conjunto de **entidades fortes**.

Considere o exemplo abaixo:

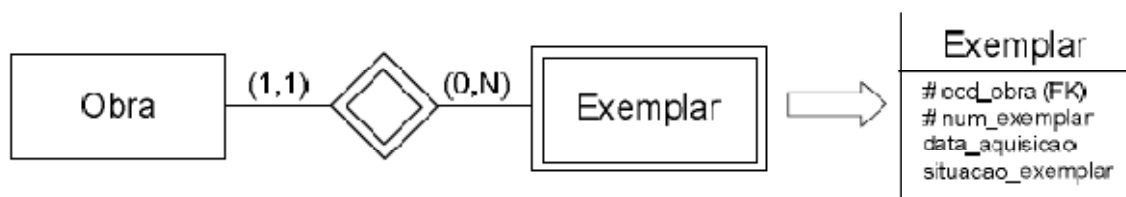



Figura 32 - Entidade Fraca

Neste exemplo fica clara a situação de modelagem chamada entidade fraca, onde a chave primária da entidade fraca (neste caso, a entidade Exemplar) é formada pela chave primária da entidade forte (no caso, a entidade Obra), mais algum atributo que diferencie seus registros (como o número do exemplar).

Nota-se assim que a entidade fraca estará sempre carregando o relacionamento com sua entidade forte, sugerindo sempre uma leitura como “um exemplar de uma determinada obra”, neste caso.

Os conceitos de conjuntos de entidades fortes e fracas estão relacionados às dependências de existência introduzidas anteriormente. Um membro de um conjunto de entidade forte é por definição uma entidade dominante, enquanto um membro de um conjunto de entidade fraca é uma entidade subordinada.

Embora um conjunto de entidades fracas não tenha uma chave primária, é necessária uma forma de distinção entre todas essas entidades no conjunto de entidades que dependa de uma entidade forte particular. O discriminador (ou chave parcial) de um conjunto de entidade fraca é um conjunto de atributos que permite que esta distinção seja feita, por exemplo, o discriminador do conjunto de entidades fracas *transação* é o atributo *número-transação*, uma vez que para cada conta um número de transação univocamente identifica uma única *transação*.

A chave primária de um conjunto de entidades fracas é formada pela chave primária do conjunto de entidades fortes do qual ele é dependente de existência (ou dependência existencial), mais seu discriminador. No caso do conjunto de entidades *transação*, sua chave primária é {*número-conta, número-transação*}, onde número conta identifica a entidade dominante de uma transação e número-transação distinguem entidades de *transação* dentro da mesma conta.

As entidades fracas são representadas por um retângulo duplicado. O conjunto de relações que identificam as entidades fracas são representadas por losângulos duplicados. Os atributos que constituem a chave parcial (ou discriminadores) são sublinhados de forma tracejada.

NOTAÇÃO:

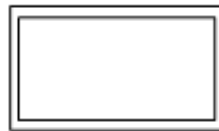



Figura 33 - Entidade Fraca

3.4.4.8. Especialização/Generalização

A **generalização** trata-se de uma abstração na qual um conjunto de entidades semelhantes, possivelmente com alguns atributos comuns e outros diferentes e com a mesma chave primária, é vistos como uma única entidade.

A **especialização** possui o mesmo conceito. A diferença está na origem das entidades.

NOTAÇÃO:

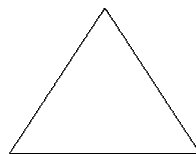



Figura 34 - Notação de Especialização /Generalização

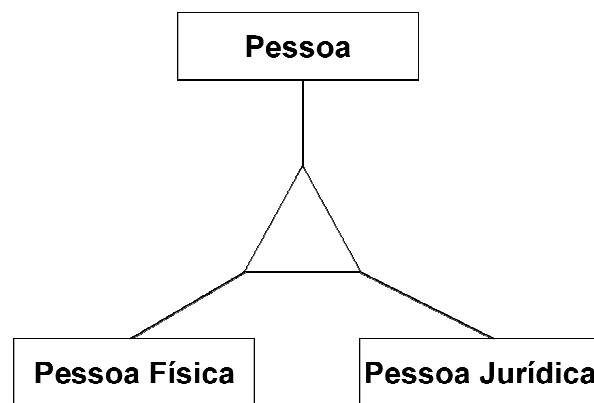



Figura 35 - Exemplo de Especialização/Generalização

Nas figuras a seguir podemos identificar com mais facilidade a diferença entre especialização e generalização. Assim, na figura 28 observa-se que a análise da representação parte do geral para o específico. Neste caso, secretária, engenheiro e motorista possuem especificações próprias, contudo todos eles são funcionário. Simplesmente foram especializados como funcionários.

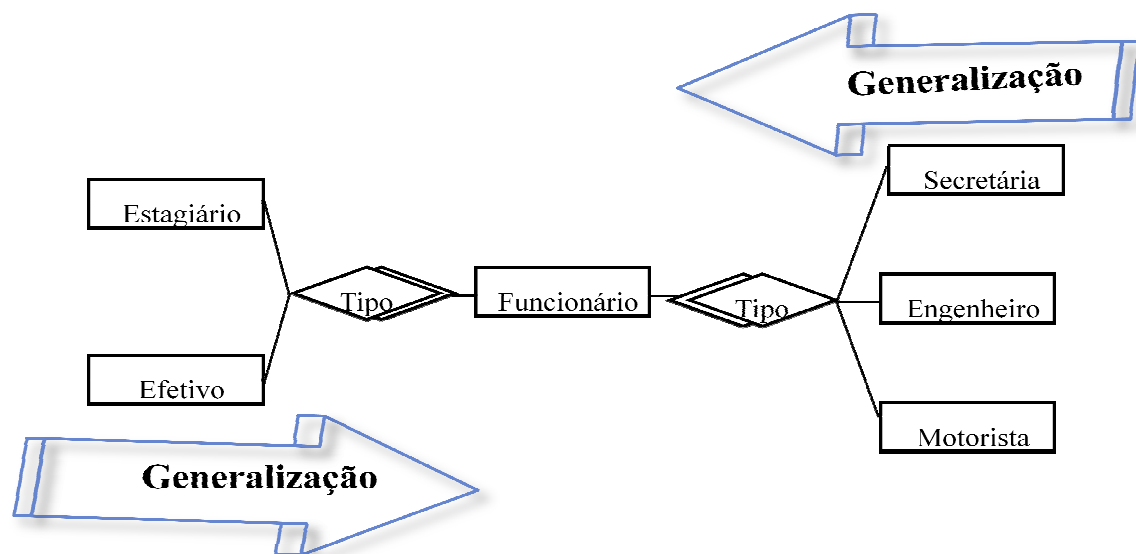



Figura 36 – Generalização

Na figura 29, observa-se a análise inversa da representação acima. Partimos do específico para o geral. Neste caso, os funcionários foram especializados em secretária, engenheiro e motorista, não perdendo suas características de funcionário, simplesmente recebendo as características peculiares de cada forma especializada

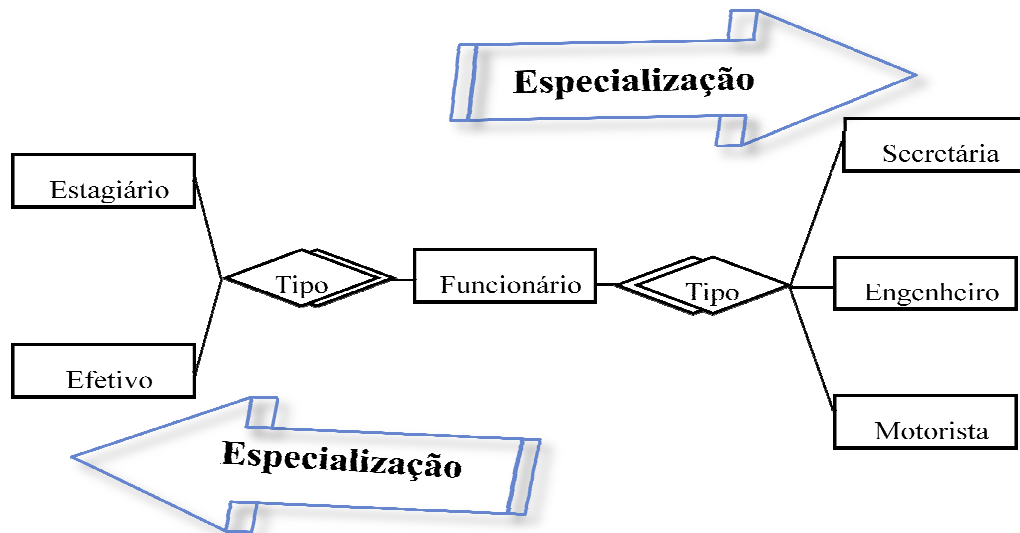



Figura 37 - Especialização

- Quando utilizar uma especialização/generalização?

Se as ocorrências de uma entidade tiverem relacionamentos ou atributos adicionais em relação às demais. O exemplo é bem claro com relação a isso: Engenheiro e Motorista podem apresentar atributos distintos como CREA e CNH, respectivamente. No entanto, ambos não deixaram de ser funcionários, somente possuem características que o outro não possui.

Uma vez descrita as estruturas básicas de um modelo entidade-relacionamento (MER) é possível compreender com facilidade um diagrama entidade-relacionamento (DER).

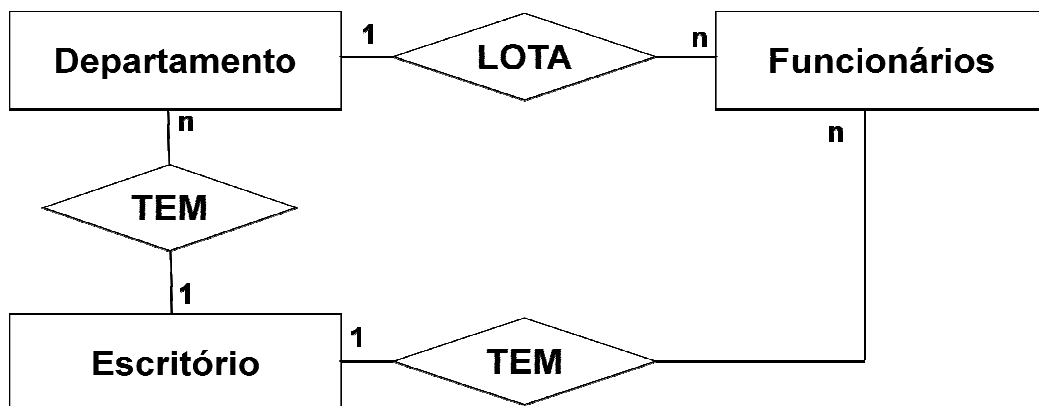



Figura 38 - Diagrama Entidade-Relacionamento

REGRAS DO MODELO CONCEITUAL

As regras do modelo conceitual visam contextualizar a utilização de recursos da Modelagem Entidade-Relacionamento ora utilizada no Modelo Conceitual. Em função do contexto é importante aplicar cada tipo de estrutura de acordo com a regra de negócio para não incorrer em inconsistências quando da utilização de uma instância do banco de dados.

4.1 MODELO CONCEITUAL COMO MODELO DE ORGANIZAÇÃO

O modelo conceitual surgiu das idéias fundamentais do projeto de banco de dados: a de que através da identificação das entidades que terão informações representadas no banco de dados é possível identificar os arquivos que comporão o banco de dados.

Na prática, convencionou-se iniciar o processo de construção de um novo banco de dados com a construção de um modelo dos objetos da organização que será atendida pelo banco de dados, ao invés de partir diretamente para o projeto do banco de dados.

Esta forma de proceder permite envolver o usuário na especificação do banco de dados. Sabe-se da prática da engenharia de software que o envolvimento do usuário na especificação do software aumenta a qualidade do software produzido. A idéia é que o usuário é aquele que melhor conhece a organização e, portanto, aquele que melhor conhece os requisitos que o software deve preencher.

Modelos conceituais são modelos que descrevem a organização e, portanto, são mais simples de compreender por usuários leigos em Informática que modelos que envolvem detalhes de implementação.

4.2 DIFERENTES MODELOS PODEM SER EQUIVALENTES

Na prática, muitas vezes observam-se analistas em acirradas discussões a fim de decidir como um determinado objeto da realidade modelada deve aparecer no modelo. Às vezes, tais discussões são absolutamente supérfluas, pois os diferentes Modelos Entidade-

Relacionamento em qualquer das opções defendidas pelos diferentes analistas geram o mesmo banco de dados. Há um conceito de equivalência entre Modelos Entidade-Relacionamento. De maneira informal, dois modelos são equivalentes quando expressam o mesmo problema, ou seja, quando modelam a mesma realidade.

Para definir o conceito de equivalência de forma mais precisa, é necessário considerar o Banco de Dados que é projetado a partir do Modelo Entidade-Relacionamento. Para fins de projeto de Banco de Dados, dois Modelos Entidade-Relacionamento são equivalentes, quando ambos geram o mesmo esquema de Banco de Dados. Assim, para analisar se dois modelos são equivalentes, é necessário considerar um conjunto de regras de tradução de Modelos Entidade-Relacionamento para modelos lógicos de banco de dados. Quando falamos "o mesmo modelo de Banco de Dados Relacional" estamos falando de bancos de dados que, abstraindo de diferenças de nomes de estruturas (tabelas, atributos, …) tenham a mesma estrutura. Um caso é o da equivalência entre um modelo que representa um conceito através de um relacionamento n:n e outro modelo que representa o mesmo conceito através de uma entidade. Um exemplo é o relacionamento CONSULTA que foi transformado em uma entidade. Os dois modelos são equivalentes, pois expressam o mesmo e geram o mesmo banco de dados.

a) A consulta como relacionamento n:n

Figura 39 - Relacionamento n:n

b) A consulta como entidade

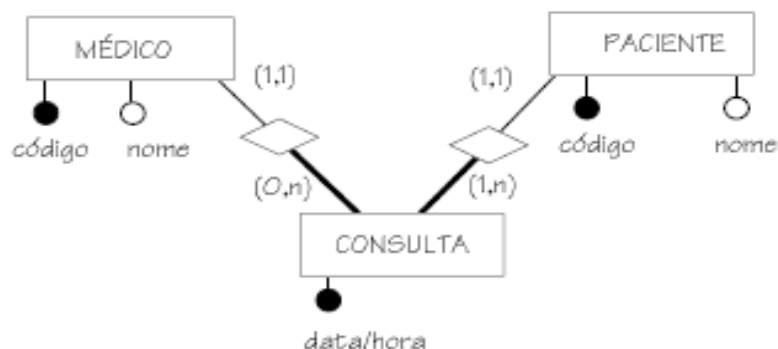



Figura 40 - Relacionamento 1:n

4.3 ATRIBUTO VERSUS ENTIDADE RELACIONADA

Uma questão que às vezes surge na modelagem de um sistema é entre modelar um objeto como sendo um atributo de uma entidade ou como sendo uma entidade autônoma relacionada a essa entidade.

Exemplificando, no caso de uma indústria de automóvel, como devemos registrar a cor de cada automóvel que sai da linha de produção? Caso considerarmos que cada automóvel possui uma única cor predominante, pode-se pensar em modelar a cor como um atributo da entidade AUTOMÓVEL ou modelar a cor como uma entidade autônoma, que está relacionada à entidade AUTOMÓVEL.

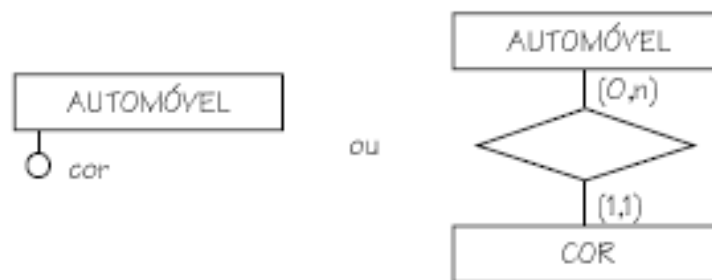



Figura 41 - Atributo X Entidade Relacionada

Alguns critérios para esta decisão são:

- Caso o objeto cuja modelagem está em discussão esteja vinculado a outros objetos (atributos, relacionamentos, entidades genéricas ou especializadas), o objeto deve ser modelado como entidade, já que um atributo não pode ter atributos, nem estar relacionado a outras entidades, nem ser generalizado ou especializado. Caso contrário, o objeto pode ser modelado como atributo. Assim, no caso do exemplo das cores dos automóveis, poder-se-ia optar por modelar a cor como uma entidade, caso se tivesse que registrar no banco de dados os possíveis fabricantes da tinta da referida cor (entidades relacionadas à cor), ou caso se quisesse registrar as datas de início e fim (atributos) do uso de uma determinada cor. Caso não houvesse nenhum objeto relacionado à cor do automóvel, poder-se-ia modelá-la como atributo da entidade automóvel.

- Quando o conjunto de valores de um determinado objeto é fixo durante toda a vida do sistema ele pode ser modelado como atributo, visto que o domínio de valores de um atributo é imutável. Quando existem transações no sistema que alteram o conjunto de valores do objeto, o mesmo não deve ser modelado como atributo. Assim, retomando o exemplo das cores dos automóveis, caso existissem transações de criação/eliminação de cores, seria preferível a modelagem de cor como entidade relacionada à entidade automóvel.

4.4 ATRIBUTO VERSUS GENERALIZAÇÃO/ESPECIALIZAÇÃO

Outro conflito de modelagem para o qual há algumas regras de cunho prático é entre modelar um determinado objeto (por exemplo, a categoria funcional de cada empregado de

uma empresa) como atributo (categoria funcional como atributo da entidade EMPREGADO) ou através de uma especialização (cada categoria funcional corresponde a uma especialização de entidade empregado).

Uma especialização deve ser usada quando se sabe que as classes especializadas de entidades possuem propriedades (atributos, relacionamentos, generalizações, especializações) particulares. Assim, no caso do exemplo acima, faz sentido especializar a entidade empregado de acordo com a categoria funcional, no caso de as classes particulares possuírem atributos ou relacionamentos próprios.

Ainda no exemplo dos empregados, o sexo do empregado é mais bem modelado como atributo de empregado, caso não existam propriedades particulares de homens e mulheres a modelar na realidade considerada.

a) Modelando categoria funcional como atributo:

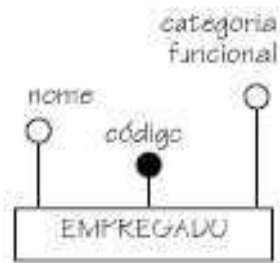



Figura 42 - Modelagem com o atributo

b) Modelando categoria funcional como especialização:

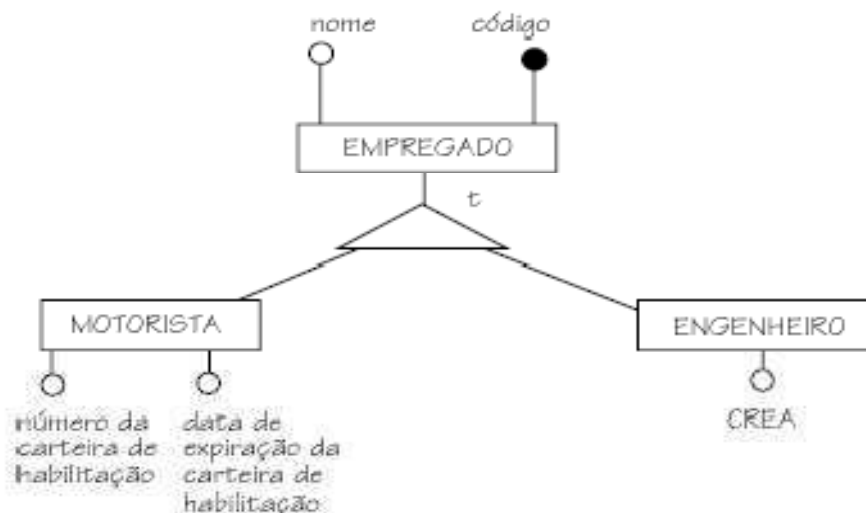



Figura 43 - Modelagem como Generalização/ Especialização

4.5 VERIFICAÇÃO DO MODELO

Uma vez confeccionado, um modelo ER deve ser validado e verificado. A verificação é o controle de qualidade que procura garantir que o modelo usado para a construção do banco de dados gerará um bom produto. Um modelo, para ser considerado bom, deve preencher uma série de requisitos, como ser completo, ser correto e não conter redundâncias.

4.5.1 O Modelo deve ser Correto

Um modelo está correto quando não contém erros de modelagem, isto é, quando os conceitos de Modelagem Entidade-Relacionamento são corretamente empregados para modelar a realidade em questão. Podem-se distinguir entre dois tipos de erros:

- Erros sintáticos
- Erros semânticos

Os **erros sintáticos** ocorrem quando o modelo não respeita as regras de construção de um Modelo Entidade-Relacionamento. Por exemplo: associar atributos a atributos; associar relacionamentos a atributos; associar relacionamentos através de outros relacionamentos ou de especializar relacionamentos ou atributos.

Os **erros semânticos** ocorrem quando o modelo, apesar de obedecer às regras de construção de Modelos Entidade-Relacionamento (estar sintaticamente correto) reflete a realidade de forma inconsistente. Por exemplo: Estabelecer associações incorretas, como associar a uma entidade um atributo que na realidade pertence à outra entidade. Isso acontece quando em um modelo com entidades CLIENTE e FILIAL, associa-se a CLIENTE o nome da FILIAL.

4.5.2 O Modelo deve ser Completo

Um modelo completo deve fixar todas as propriedades desejáveis do banco de dados. Isso obviamente somente pode ser verificado por alguém que conhece profundamente o sistema a ser implementado. Uma boa forma de verificar se o modelo é completo é verificar se todos os dados que devem ser obtidos do banco de dados estão presentes e se todas as transações de modificação do banco de dados podem ser executadas sobre o modelo.

Este requisito é aparentemente conflitante com a falta de poder de expressão de Modelos Entidade-Relacionamento. Quando dizemos que um modelo deve ser completo, estamos exigindo que todas as propriedades expressáveis no modelo conceitual apareçam no modelo.

4.5.3 Modelo Deve ser Livre de Redundâncias

Um modelo deve ser mínimo, isto é não deve conter conceitos redundantes. Um tipo de redundância que pode aparecer é a de relacionamentos redundantes. Relacionamentos

redundantes são relacionamentos que são resultado da combinação de outros relacionamentos entre as mesmas entidades.

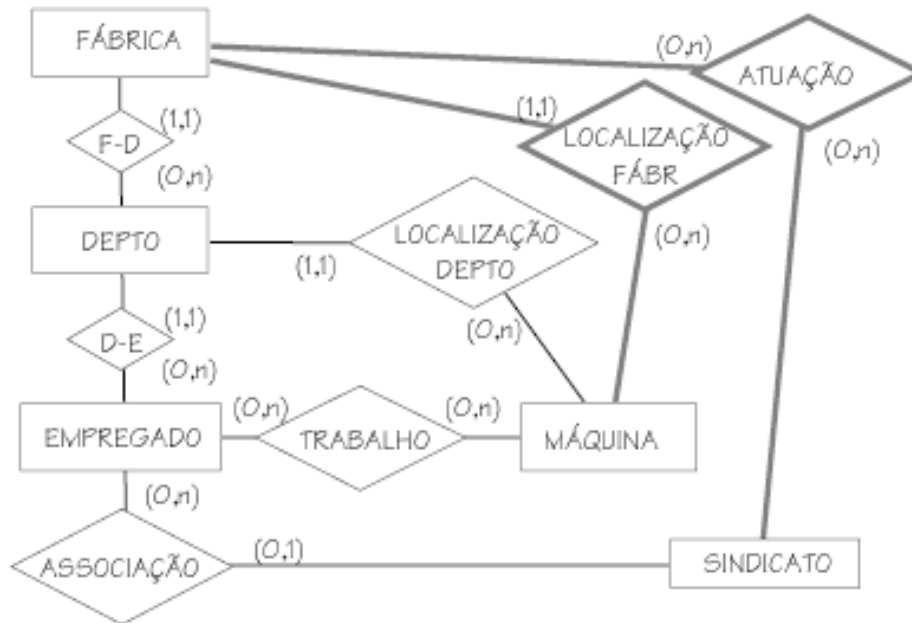



Figura 44 - Modelo Redundante

4.5.4 Modelo Deve Refletir o Aspecto Temporal

Usualmente, ao iniciar a modelagem Entidade-Relacionamento, a preocupação é obter um modelo que descreva os estados válidos e corretos do banco de dados. O primeiro modelo tende a refletir um estado momentâneo do banco de dados. Entretanto, é necessário lembrar que assim como informações são incluídas no banco de dados, elas também podem ter que ser eliminadas do banco de dados.

Um banco de dados não pode crescer indefinidamente. Informações ultrapassadas ou desnecessárias podem ser eliminadas. Portanto, é necessário considerar o aspecto temporal na modelagem de dados. Não há regras gerais como proceder neste caso, mas é possível identificar alguns padrões que se repetem freqüentemente na prática.

a) Manter ou não um histórico de salários?

Figura 45 - Manutenção de histórico de salário

b) Manter ou não o histórico de alocação de mesas?

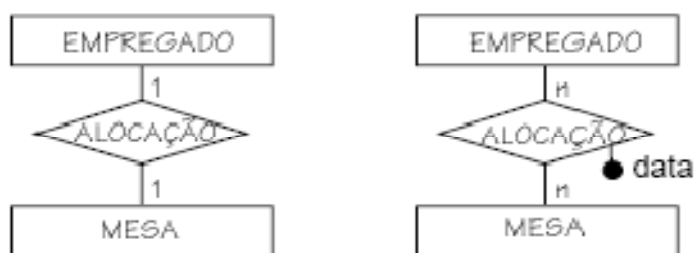



Figura 46 - Manutenção de histórico de empregado-mesa

c) Manter ou não o histórico de lotação do empregado?

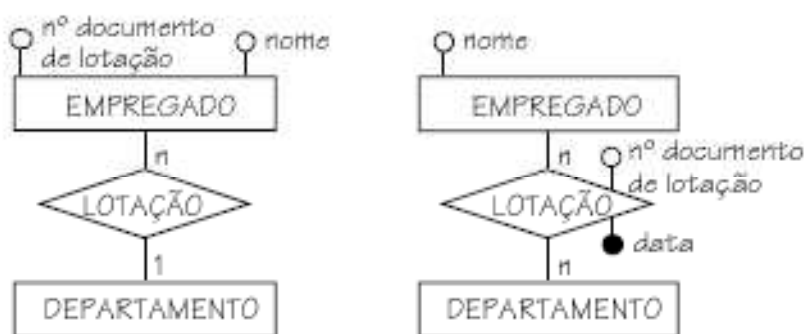



Figura 47 - Manutenção de histórico de empregado-departamento

d) Manter somente a inscrição atual ou o histórico de todas as inscrições nos cursos?

Figura 48 - Manutenção de histórico de participante

Parte
5

ABORDAGEM RELACIONAL

A abordagem relacional é muito próxima do modelo lógico, é uma descrição de um banco de dados no nível de abstração visto pelo usuário do Sistema Gerenciador de Banco de Dados - SGBD. Assim, o modelo lógico é dependente do tipo particular de SGBD que está sendo usado.

Observe o modelo conceitual abaixo:

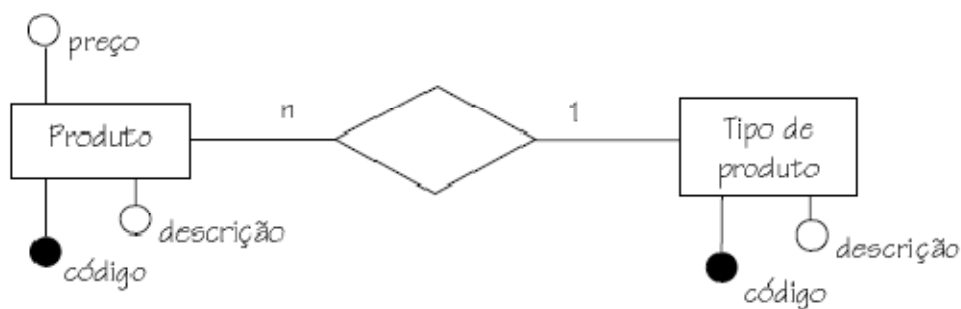



Figura 49 - Modelo Conceitual

A próxima figura mostra um exemplo de Banco de Dados relacional projetado a partir do modelo conceitual mostrado na figura acima.

TipoDeProduto

CodTipoProd	DescrTipoProd
1	Computador
2	Impressora

Produto

CodProd	DescrProd	PrecoProd	CodTipoProd
1	PC desktop modelo X	2.500	1
2	PC notebook ABC	3.500	1
3	Impressora jato de tinta	600	2
4	Impressora laser	800	2

Figura 50 - Modelo Lógico

Assim sendo, podemos concluir que um modelo lógico para o Banco de Dados acima deve definir quais as tabelas que o banco contém e, para cada tabela, quais os nomes das colunas.

Essa estrutura também pode ser representada da seguinte forma:

```
TipoDeProduto(CodTipoProd,DescrTipoProd)
Produto(CodProd,DescrProd,PrecoProd,CodTipoProd)
CodTipoProd referencia TipoDeProduto
```

5.1 COMPOSIÇÃO DE UM MODELO CONCEITUAL

Um banco de dados relacional é composto de tabelas ou relações.

5.1.1 Tabela

Uma tabela é um conjunto não ordenado de linhas (tuplas, na terminologia acadêmica). Cada linha é composta por uma série de campos (valor de atributo, na terminologia acadêmica). Cada campo é identificado por nome de campo (nome de atributo, na terminologia acadêmica). O conjunto de campos das linhas de uma tabela que possuem o mesmo nome formam uma coluna.

CódigoEmp	Nome	CodigoDepto	CategFuncional
E5	Souza	D1	C5
E3	Santos	D2	C5
E2	Silva	D1	C2
E1	Soares	D1	—

Figura 51 - Tabela

5.1.2 Chaves

O conceito básico para estabelecer relações entre linhas de tabelas de um banco de dados relacional é o da chave. Em um banco de dados relacional, há ao menos dois tipos de chaves a considerar: a chave primária e a chave estrangeira.

a) **Chave Primária:** é uma coluna ou uma combinação de colunas cujos valores distinguem uma linha das demais dentro de uma tabela.

Dependente

CódigoEmp	NoDepen	Nome	Tipo	DataNasc
E1	01	João	Filho	12/12/91
E1	02	Maria	Esposa	01/01/50
E2	01	Ana	Esposa	05/11/55
E5	01	Paula	Esposa	04/07/60
E5	02	José	Filho	03/02/85

Figura 52 - Chave Primária

b) **Chave Estrangeira:** é uma coluna ou uma combinação de colunas, cujos valores aparecem necessariamente na chave primária de uma tabela. A chave estrangeira é o mecanismo que permite a implementação de relacionamentos em um banco de dados relacional.

Dept

CodigoDepto	NomeDepto
D1	Compras
D2	Engenharia
D3	Vendas

Emp

CodigoEmp	Nome	CodigoDepto	CategFuncional	CIC
E1	Souza	D1	-	132.121.331-20
E2	Santos	D2	C5	891.221.111-11
E3	Silva	D2	C5	341.511.775-45
E5	Soares	D1	C2	631.692.754-88

Figura 53 - Chave Estrangeira

A existência de uma chave estrangeira impõe restrições que devem ser garantidas em diversas situações de alteração do banco de dados:

- Quando da inclusão de uma linha na tabela que contém a chave estrangeira, deve ser garantido que o valor da chave estrangeira apareça na coluna da chave primária referenciada. No exemplo acima, isso significa que um novo empregado deve atuar em um departamento já existente no banco de dados.

- Quando da alteração do valor da chave estrangeira deve ser garantido que o novo valor de uma chave estrangeira apareça na coluna da chave primária referenciada.

- Quando da exclusão de uma linha da tabela que contém a chave primária referenciada pela chave estrangeira deve ser garantido que na coluna chave estrangeira não apareça o valor da chave primária que está sendo excluída. No exemplo acima, isso significa que um departamento não pode ser excluído, caso nele ainda existirem empregados.

5.1.3 Domínios e valores vazios

Quando uma tabela do banco de dados é definida, para cada coluna da tabela, deve ser especificado um conjunto de valores (alfanumérico, numérico,…) que os campos da respectiva coluna podem assumir. Este conjunto de valores é chamado de domínio da coluna ou domínio do campo.

Além disso, devem ser especificados se os campos das colunas podem estar vazios (“null” em inglês) ou não. Estar vazio indica que o campo não recebeu nenhum valor de seu domínio.

As colunas nas quais não são admitidos valores vazios são chamadas de colunas obrigatórias (“*not null*”). As colunas nas quais podem aparecer campos vazios são chamadas de colunas opcionais(“*null*”).

Normalmente, os SGBD relacionais exigem que toda a coluna que compõem a chave primária seja obrigatória. A mesma exigência não é feita para as demais chaves.

5.1.4 Restrições de Integridade

Um dos objetivos primordiais de um SGBD é a integridade de dados. Dizer que os dados de um banco de dados estão íntegros significa dizer que eles refletem corretamente a realidade representada pelo banco de dados e que são consistentes entre si. Para tentar garantir a integridade de um banco de dados os SGBD oferecem o mecanismo de restrições de integridade. Uma restrição de integridade **é uma regra de consistência de dados que é garantida pelo próprio SGBD.** As restrições de integridade classificam-se em:

- Integridade de domínio: Restrições deste tipo especificam que o valor de um campo deve obedecer a definição de valores admitidos para a coluna (o domínio da coluna). Nos SGBD relacionais comerciais, é possível usar apenas domínios pré-definidos (número inteiro, número real, alfanumérico de tamanho definido, data, …). O usuário do SGBD não pode definir domínios próprios de sua aplicação (por exemplo, o domínio dos dias da semana ou das unidades da federação).

- Integridade de vazio: Através deste tipo de restrição de integridade é especificado se os campos de uma coluna podem ou não ser vazios (se a coluna é obrigatória ou opcional). Como visto, os campos que compõem a chave primária sempre devem ser diferentes de vazio.

- Integridade de chave: Trata-se da restrição que define que os valores da chave primária e alternativa devem ser únicos.

- Integridade referencial: É a restrição que define que os valores dos campos que aparecem em uma chave estrangeira devem aparecer na chave primária da tabela referenciada.

As restrições dos tipos acima especificados devem ser garantidas automaticamente por um SGBD relacional.

5.2 ESPECIFICAÇÃO DE BANCO DE DADOS RELACIONAIS

A especificação de um banco de dados relacional (chamada de esquema do banco de dados) deve conter no mínimo a definição do seguinte:

- Tabelas que formam o banco de dados
- Colunas que as tabelas possuem
- Restrições de integridade

Na prática, na definição de esquemas relacionais são usadas diversas notações, que variam de um SGBD para o outro.

Parte
6

TRANSFORMAÇÃO ENTRE MODELOS

Até aqui foram vistas duas formas de modelagem de dados: a Abordagem Entidade-Relacionamento e a Abordagem Relacional. Estas abordagens propõem modelar os dados em diferentes níveis de abstração. A Abordagem Entidade-Relacionamento é voltada à modelagem de dados de forma independente do SGBD considerado. É adequada para construção do modelo conceitual. Já a Abordagem Relacional modela os dados em nível de SGBD relacional. Um modelo neste nível de abstração é chamado de Modelo Lógico.

A figura abaixo dá uma visão geral dos processos de transformação entre modelos.

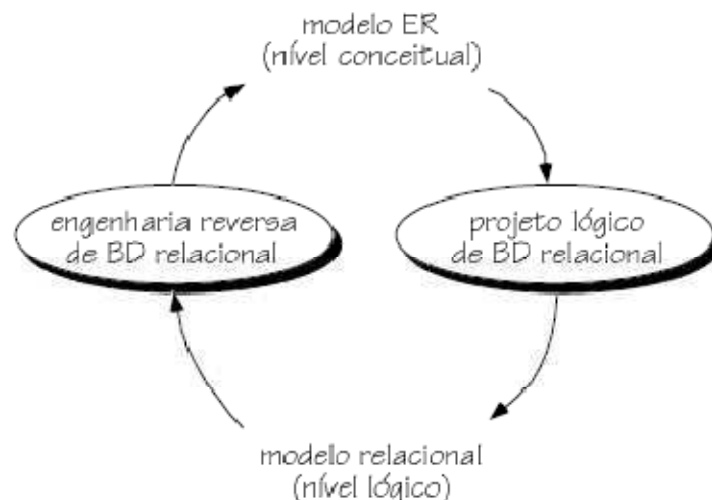



Figura 54 - Transformação entre Modelos

6.1 TRANSFORMAÇÃO ENTIDADE-RELACIONAMENTO PARA RELACIONAL

As regras foram definidas tendo em vista dois objetivos básicos:

- Obter um banco de dados que permita boa performance de instruções de consulta e alteração do banco de dados. Obter boa performance significa basicamente diminuir o número

de acessos a disco, já que estes consomem o maior tempo na execução de uma instrução de banco de dados.

- Obter um banco de dados que simplifique o desenvolvimento e a manutenção de aplicações.

A fim de alcançar estes objetivos, as regras de tradução foram definidas tendo por base, entre outros, os seguintes princípios:

- Evitar junções
- Diminuir o número de Chaves Primárias
- Evitar campos opcionais (campos *null*)

Observe o modelo abaixo:

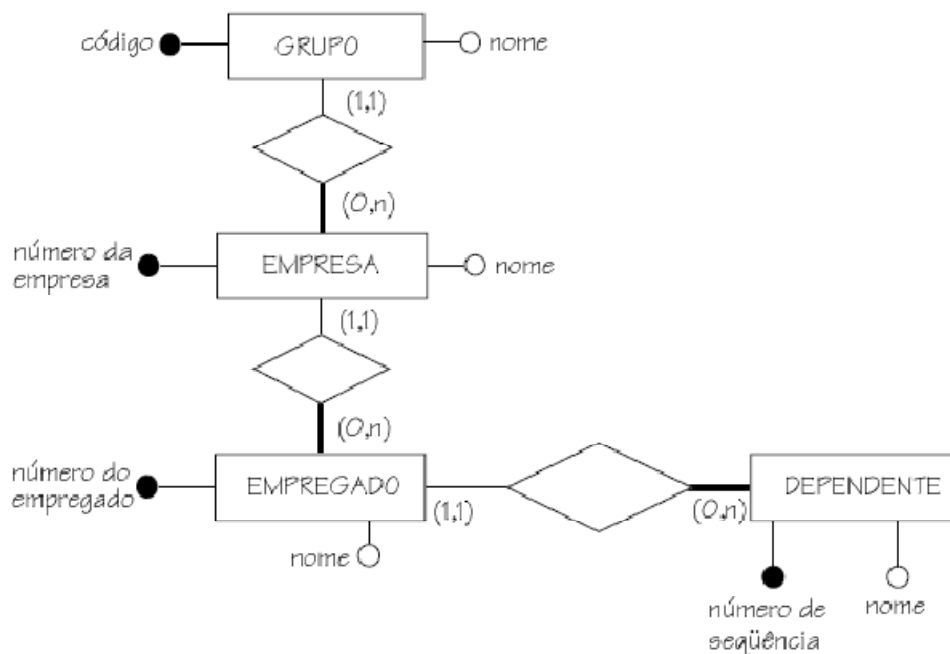



Figura 55 - Modelo Conceitual

6.1.1 Implementação de Relacionamentos

Na conversão dos relacionamentos do Modelo Entidade-Relacionamento para o Modelo Lógico é importante observar o lado que recebe o relacionamento o **N**, visto que ele determinará em que tabela/entidade ficará a chave estrangeira. A representação da relação Empregado e Dependente no modelo lógico usando a ferramenta DBDesign ficaria da seguinte forma:

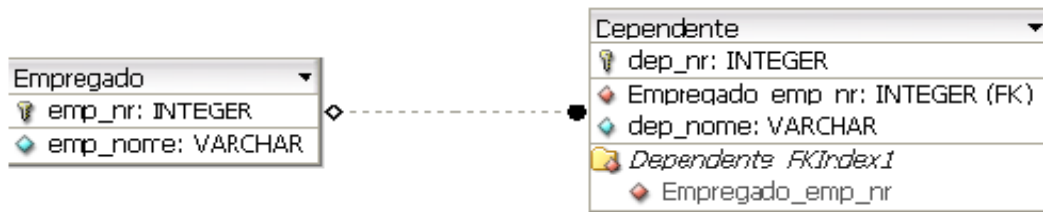



Figura 56 – Representação 1:n no DBDesign

Caso o relacionamento seja N:N como acontece na relação Engenheiro e Projeto a representação no modelo conceitual seria da seguinte forma:

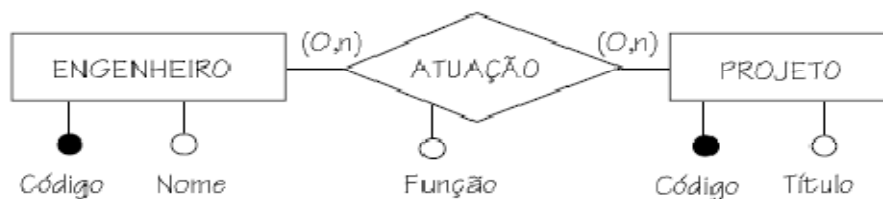



Figura 57 - Representação Conceitual

No modelo lógico a representação através da ferramenta DBDesign seria da seguinte forma:

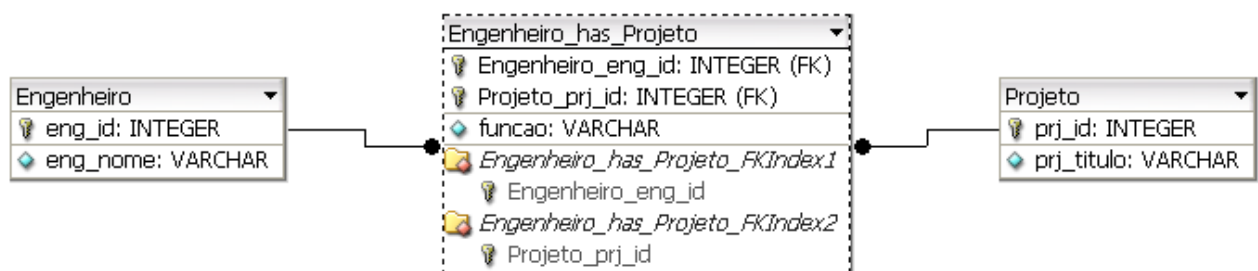
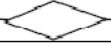
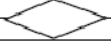
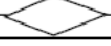
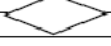



Figura 58 - Representação n:n no DBDesign

Observa-se que para uma relação N:N, há a necessidade de se criar uma nova tabela que conterá as chaves estrangeiras de Engenheiro e de Projeto, podendo, neste caso, conter outros atributos como é o caso do atributo função.

Além desses casos existem outros onde há a necessidade da criação de uma nova tabela, contudo dependerá da análise realizada sobre o problema. Para isso, é conveniente conhecer as regras expressas no quadro abaixo:

Tipo de relacionamento	Regra de implementação		
	Tabela própria	Adição coluna	Fusão tabelas
Relacionamentos 1:1			
(0,1) ◇ (0,1)	±	✓	×
(0,1) ◇ (1,1)	×	±	✓
(1,1) ◇ (1,1)	×	±	✓
Relacionamentos 1:n			
(0,1) ◇ (0,n)	±	✓	×
(0,1) ◇ (1,n)	±	✓	×
(1,1) ◇ (0,n)	×	✓	×
(1,1) ◇ (1,n)	×	✓	×
Relacionamentos n:n			
(0,n ◇ (0,n)	✓	×	×
(0,n ◇ (1,n)	✓	×	×
(1,n) ◇ (1,n)	✓	×	×

✓ Alternativa preferida ± Pode ser usada × Não usar

Para relacionamentos especializados/generalizados é importante notar que é mantida a mesma chave para as tabelas envolvidas no relacionamento em relação a tabela principal.

Observe a figura a seguir:

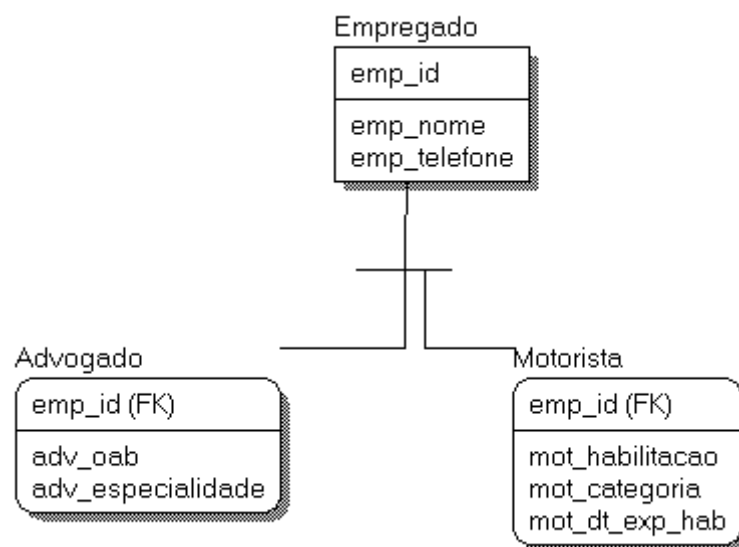



Figura 59 - Relacionamento de Especialização/Generalização

Parte
7

NORMALIZAÇÃO

Nas sessões anteriores foi possível compreender como se dá a analise de requisitos de um negócio, a conseqüente formatação de um banco de dados usando a abordagem entidade-relacionamento e sua transformação para Modelo Lógico.

Ainda dentro do Modelo Lógico de dados é importante saber que a criação de um banco de dados não advém somente da análise de requisitos, podendo se formar a partir de uma coleção de dados armazenados de forma organizada, porém não informatizado. Assim sendo, é importante para o analista ou DBA saber aproveitar esta organização e construir modelo de dados baseado no que já existe.

Dentro deste novo conceito de modelagem dos dados é importante rever nas seções anteriores os conceitos de independência de dados e de consistência de dados. Neste último, destaca-se o fato da consistência de dados poder ser obtida de diversas formas, a saber:

- Pelo Sistema Gerenciador de Banco de Dados;

- Pelos aplicativos; e

- Pela própria construção do sistema.

Pelo próprio Sistema Gerenciador de Banco de Dados é oferecida consistência através de algumas regras de integridade que ele mesmo implementa, garantindo, por exemplo, validade e completeza das informações.

As regras mais importantes oferecidas pelo Sistema Gerenciador de Banco de Dados são:

- Integridade de domínio;

- Integridade de Chave;

- Integridade de Vazio;

- Integridade referencial.

SGBD → Regras de Integridade
Validade **Completeza**

Pelos aplicativos, consideramos que nada foi implementado no Sistema Gerenciador de Banco de Dados. Assim sendo, é indispensável que sejam criadas rotinas para garantia de uma consistência mínima dos dados, como por exemplo, o respeito a domínios de chave estrangeira.

Pela própria construção do sistema, que é o que nos interessa neste momento. Por este método é necessário controlar a construção do sistema através da criação de tabelas segundo regras que garantam a manutenção de certas propriedades. Assim, as tabelas que atendem a um determinado conjunto de regras, diz-se estarem em uma determinada **forma normal**.

7.1 DEPENDÊNCIA FUNCIONAL

O conceito de dependência funcional é muito importante para o entendimento da normalização. Uma dependência funcional é uma restrição entre dois conjuntos de atributos de uma base de dados.

Se o valor de um atributo ou conjunto de atributos **A** permite descobrir o valor de outro atributo ou conjunto de atributos **B**, dizemos que **A** determina funcionalmente **B**, ou que **B** depende de **A**, e denotamos:

Figura 60 - Representação da Dependência Funcional

Exemplos:

RG → { Nome, CIC, Depto., RG_Supervisor, Salário }

Número_Projeto → { Nome_Projeto, Localização }

{ RG_Empregado, Número_Projeto } → **Horas**

Observe as tabelas abaixo:

Matrícula	Nome	Endereço	...	...	...	...
85001	Pedro	Av. D1, 25				
85001	Pedro	Av. D1, 25				
85001	Pedro	Av. D1, 25				
86005	Ana	Av. C-4, 35				

Dependência funcional de matrícula

...	...	...	Disciplina	Descrição	Instrutor	...
			CP302	Banco de Dados	Nestor	
			CP303	Comunicações	Ana	
			CP304	Eng. Software	Jorge	
			CP302	Banco de Dados	Nestor	

Dependência funcional de disciplina

...	...	...	...	...	Instrutor	Sala
					Nestor	102
					Ana	500
					Jorge	102
					Nestor	1024

Dependência funcional de instrutor

7.2 NORMALIZAÇÃO

O processo de normalização pode ser visto como o processo no qual são eliminados esquemas de relações (tabelas) não satisfatórias, decompondo-as, através da separação de seus atributos em esquemas de relações menos complexas, mas que satisfaçam as propriedades desejadas.

O processo de normalização como foi proposto inicialmente por Codd. Esse processo conduz um esquema de relação através de uma bateria de testes para certificar se o mesmo

encontra-se na 1ª, 2ª e 3ª Formas Normais. Essas três Formas Normais são baseadas nas dependências funcionais dos atributos da relação.

7.2.1 Objetivo da Normalização

- Evitar problemas provocados por falhas no projeto, bem como eliminar a "mistura de assuntos" e repetições desnecessárias de dados.

- Uma Regra de Ouro que devemos observar quando do Projeto de um Banco de Dados baseado no Modelo Relacional de dados é a de "não misturar assuntos em uma mesma tabela".

O Processo de Normalização aplica uma série de Regras sobre as Tabelas de um Banco de Dados, para verificar se estas estão corretamente projetadas. Embora existam cinco formas normais (ou regras de Normalização), na prática usamos um conjunto de três Formas Normais.

Normalmente após a aplicação das Regras de Normalização, algumas tabelas acabam sendo divididas em duas ou mais tabelas, o que no final gera um número maior de tabelas do que o originalmente existente.

Este processo causa a simplificação dos atributos de uma tabela, colaborando significativamente para a estabilidade do modelo de dados, reduzindo-se consideravelmente as necessidades de manutenção.

7.2.2 Vantagens da Normalização

- Otimiza o desempenho das atualizações;

- Agrupa dados de tal forma que numa Relação cada dado torna-se um dado dependente da Chave, de toda a Chave e nada mais que a Chave primária da Relação;

- Provê um meio formal de se estruturar a informação, de tal forma, que fica claro, que tipos de dados existem e que dependências funcionais são satisfeitas.

- Ajuda a dirimir dúvidas de Projeto, pois por vezes, com a finalidade de melhorar o desempenho das funções de acesso, o projetista do Banco de Dados pode tomar decisões que comprometam as funções de atualização. Ao seguir estritamente o processo de normalização isso seria evitado;

7.2 FORMAS NORMAIS

7.2.1 Primeira Forma Normal (1ª FN)

- Uma Tabela está na Primeira Forma Normal quando seus atributos não contêm grupos de repetição;

- Todos os atributos de uma tabela devem ser atômicos (indivisíveis), ou seja, não são permitidos atributos multivalorados, atributos compostos ou atributos multivalorados compostos.

- Tem por objetivo corrigir registros em que, para a chave primária, ocorrem "grupos de repetição", ou seja, atributos que possam assumir múltiplos valores.

Regra:

Se um depósito de dados apresentar grupos de repetição, desmembrá-lo, criando depósitos menores e sem grupo de repetição, ou separar tabelas aninhadas.

Exemplo:

CodPesq	Tipo	Desc	Matr	Nome	Cat	Cred	DtMatr	Duracao
MA001	Mat. Aplic	Informática	5645	Mario	M1	20	12/03/00	10
			5963	Carlos	M2	30	15/02/00	12
			8942	Manoel	G3	17	25/04/00	16
			1787	Joaquim	D4	60	07/01/00	30
			4893	Osmar	D3	54	25/03/00	28
INF003	Sist. Lin	Redes Neurais	4893	Osmar	D3	54	25/03/00	28
			5645	Mario	M1	20	12/03/00	10
			8942	Manoel	G3	17	25/04/00	16

Tabela aninhada

Pesquisa

CodPesq	Tipo	Desc
MA001	Mat. Aplic	Informática
INF003	Sist. Lin	Redes Neurais

Aluno

CodPesq	Matr	Nome	Cat	Cred	DtMatr	Duracao
MA001	5645	Mario	M1	20	12/03/00	10
MA001	5963	Carlos	M2	30	15/02/00	12
INF003	8942	Manoel	G3	17	25/04/00	16
INF003	1787	Joaquim	D4	60	07/01/00	30
INF003	4893	Osmar	D3	54	25/03/00	28

Figura 61 - 1ª Forma Normal

7.2.1 Segunda Forma Normal (2ª FN)

- Uma Tabela está na Segunda Forma Normal quando, além de encontrar-se na primeira forma normal, cada coluna não chave depende da chave primária completa.

- Tem por objetivo corrigir registros de chave composta que tenham atributos dependentes apenas de uma parte da chave.

Regra:

Se um depósito de dados apresentar atributos dependentes apenas de uma parte da chave composta, desmembrá-lo, criando depósitos menores em que os atributos sejam dependentes de toda a chave.

Obs.:

Se na 1ª Forma Normal a tabela possuir apenas uma coluna como chave primária, então ela já está na 2ª Forma Normal.

Exemplo:

Aluno

CodPesq	Matr	Nome	Cat	Cred	DtMatr	Duracao
MA001	5645	Mario	M1	20	12/03/00	10
MA001	5963	Carlos	M2	30	15/02/00	12
INF003	8942	Manoel	G3	17	25/04/00	16
INF003	1787	Joaquim	D4	60	07/01/00	30
INF003	4893	Osmar	D3	54	25/03/00	28

Dependência Parcial

Pesquisa

CodPesq	Tipo	Desc
MA001	Mat. Aplic	Informática
INF003	Sist. Lin	Redes Neurais

PesqAluno

CodPesq	Matr	DtMatr	Duracao
MA001	5645	12/03/00	10
MA001	5963	15/02/00	12
INF003	8942	25/04/00	16
INF003	1787	07/01/00	30
INF003	4893	25/03/00	28

Aluno

Matr	Nome	Cat	Cred
5645	Mario	M1	20
5963	Carlos	M2	30
8942	Manoel	G3	17
1787	Joaquim	D4	60
4893	Osmar	D3	54

A Segunda Forma normal está ligada a ocorrência de **Chaves Primárias compostas.**

Figura 62 - 2ª Forma Normal

7.2.3 Terceira Forma Normal (3ª FN)

- Encontra-se na segunda forma normal; e

- Na definição dos campos de uma entidade podem ocorrer casos em que um campo não seja dependente diretamente da chave primária ou de parte dela, mas sim dependente de outro campo da tabela, campo este que não seja a Chave Primária.

- Tem por objetivo corrigir registros que apresentem atributos dependentes de um atributo não chave.

- Uma coluna não chave primária depende funcionalmente de outra coluna ou combinação de colunas não chave primária.

- Eliminam-se campos calculados.

Regra:

Se um depósito de dados tiver atributo dependente de um atributo não chave, desmembrá-lo, criando depósitos menores em que os atributos só dependam da chave primária.

Exemplo:

Aluno

Matr	Nome	Cat	Cred
5645	Mario	M1	20
5963	Carlos	M2	30
8942	Manoel	G3	17
1787	Joaquim	D4	60
4893	Osmar	D3	54

Dependência transitivas

PesqAluno

CodPesq	Matr	DtMatr	Duracao
MA001	5645	12/03/00	10
MA001	5963	15/02/00	12
INF003	8942	25/04/00	16
INF003	1787	07/01/00	30
INF003	4893	25/03/00	28

Aluno

Matr	Nome	Cat
5645	Mario	M1
5963	Carlos	M2
8942	Manoel	G3
1787	Joaquim	D4
4893	Osmar	D3

Cat

Cat	Cred
M1	20
M2	30
G3	17
D4	60
D3	54

Pesquisa

CodPesq	Tipo	Desc
MA001	Mat. Aplic	Informática
INF003	Sist. Lin	Redes Neurais

Figura 63 - 3ª Forma Normal

7.2.4 Demais Formas Normais

Para a maioria dos documentos e arquivos, a decomposição até a 3ª Forma Normal é suficiente para obter o esquema de um banco de dados correspondente ao documento. Na literatura encontram-se outras formas normais, como a forma normal de Boyce/Codd, a 4ª Forma Normal e a 5ª Forma Normal.

A **Quarta Forma Normal** é empregada se ainda existir alguma redundância na 3ª Forma Normal. Isso acontecerá quando um atributo não chave contiver valores múltiplos para uma mesma chave.

A **Quinta Forma Normal** é um caso muito raro de ocorrer. É como se tivéssemos na 4ª Forma Normal três campos em uma tabela e esses campos tivessem valores multivalorados.

A **Forma Normal Boyce/Codd** é um aperfeiçoamento da 3ª Forma Normal e serve para impor a condição de que uma tabela contenha dependência apenas em função de chave(s) candidata(s). Qualquer outra dependência deve ser eliminada e transferida para outra tabela onde o determinante seja chave candidata.

ESTRATÉGIAS DE PROJETO DE BANCO DE DADOS

Uma vez conhecedores dos conceitos do modelo conceitual e das duas estratégias do modelo lógico, é possível traçar um projeto de banco de dados alicerçado nos conceitos aprendidos. O projeto de banco de dados é parte integrante do desenvolvimento de um sistema de informação. Nesta fase, uma das preocupações é com a correta representação dos dados operacionais. A atividade de projetar banco de dados envolve a definição de esquemas de dados em diferentes níveis de abstração: Nível Conceitual, Lógico e Físico.

Projetar um banco de dados envolve basicamente duas metodologias principais: ***Top-Down*** e ***Bottom-Up***. A primeira, parte da representação de mais alto nível de abstração para a de mais baixo nível de abstração, e a segunda parte da representação de mais baixo nível de abstração para a mais alta.

8.1 PROJETO *TOP-DOWN* DE BANCO DE DADOS

Nesta metodologia a ênfase está nos requisitos da aplicação que são obtidos com o usuário e através da compreensão dos dados operacionais relevantes para a aplicação.

Este é o processo mais usual, pois é aplicado onde não existe sistema informatizado ou banco de dados anterior.

Envolve quatro etapas:

1. Levantamento de requisitos
2. Projeto Conceitual
3. Projeto lógico
4. Projeto físico ou implementação

8.1.1 Levantamento de Requisitos

Envolve a coleta de informações sobre os dados, suas restrições e seus relacionamentos na organização.

Sua forma de realização contempla reuniões com os usuários e observação do funcionamento da organização.

O resultado é um documento com a especificação dos requisitos que podem ser realizados de forma narrativa ou itemizado.

Exemplo:

LEVANTAMENTO NARRATIVO

... Todo servidor possui uma identificação única na universidade e está lotado em um departamento onde exerce uma determinada função.

LEVANTAMENTO ITEMIZADO

a) Servidor:

- Possui uma identificação única na faculdade;

- Está lotado em um departamento;

- Exerce uma função no departamento.

8.1.2 Projeto Conceitual

O Projeto conceitual contempla a Modelagem dos dados e seus relacionamentos independentes da estrutura de representação do Sistema Gerenciador de Banco de Dados.

Sua forma de realização se dá através da análise da especificação de requisitos.

O resultado é um esquema conceitual, proporcionando a abstração de dados de alto nível, uma fácil compreensão pelo usuário leigo, fácil manutenção dos dados possibilitando a tradução para qualquer Sistema Gerenciador de Banco de Dados;

8.1.3 Projeto Lógico

O Projeto Lógico é a conversão do esquema conceitual para um esquema de representação de um Sistema Gerenciador de Banco de Dados (esquema lógico).

Sua forma de realização é pela aplicação de regras de conversão entre modelos.

O resultado é o diagrama lógico (tabelas, restrições, etc). Nesta etapa é recomendável utilizar-se de uma ferramenta *case* para modelagem do diagrama.

8.1.4 Projeto Físico

O projeto físico é a definição do esquema lógico em um Sistema Gerenciador de Banco de Dados adequado ao modelo.

Sua forma de realização é a criação do script SQL. Esse script pode ser gerado a partir de uma ferramenta *case*, a mesma usada na etapa anterior.

O Resultado é um esquema físico ou script SQL pronto para ser aplicado no Sistema Gerenciador de Banco de Dados.

8.1.5 Objetivos do Projeto *Top-Down*

- Projeto Conceitual: Correta abstração do mundo real (captura correta da semântica da aplicação)

- Projeto Lógico e Físico: Escolhas corretas na conversão para o esquema do SGBD (relacional) para maximizar a performance de acesso (distribuição adequada dos dados na tabela)

8.2 PROJETO *BOTTOM-UP* DE BANCO DE DADOS

Esta estratégia tem ênfase nas descrições de dados já existentes na organização, como arquivos eletrônicos, fichários, pedidos, Notas Fiscais, etc.

É também chamado de um processo de engenharia reversa e é aplicado em casos em que existem fontes de dados ou sistemas informatizados (legados) sem Banco de Dados.

Envolve cinco etapas:

1. Coleta da fonte de dados;
2. Representação em uma tabela não normalizada;
3. Normalização;
4. Integração de esquemas relacionais/Projeto Lógico;
5. Engenharia Reversa/Projeto físico ou implementação

8.2.1 Coleta da Fonte de Dados

Esta fase busca por fontes que organizam dados operacionais de alguma maneira, como arquivos, fichários, relatórios, etc.

Exemplo:

Um relatório de alocação de projeto

RELATÓRIO DE ALOCAÇÃO A PROJETO

CÓDIGO DO PROJETO: LSC001 TIPO: Novo Desenv.
DESCRIÇÃO: Sistema de Estoque

CÓDIGO DO EMPREGADO	NOME	CATEGORIA FUNCIONAL	SALÁRIO	DATA DE INÍCIO NO PROJETO	TEMPO ALOCADO AO PROJETO
2146	João	A1	4	1/11/91	24
3145	Silvio	A2	4	2/10/91	24
6126	José	B1	9	3/10/92	18
1214	Carlos	A2	4	4/10/92	18
8191	Mário	A1	4	1/11/92	12

CÓDIGO DO PROJETO: PAG02 TIPO: Manutenção
DESCRIÇÃO: Sistema de RH

CÓDIGO DO EMPREGADO	NOME	CATEGORIA FUNCIONAL	SALÁRIO	DATA DE INÍCIO NO PROJETO	TEMPO ALOCADO AO PROJETO
8191	Mário	A1	4	1/05/93	12
4112	João	A2	4	4/01/91	24
6126	José	B1	9	1/11/92	12

Figura 64 - Coleta de Fontes

8.2.2 Representação em uma Tabela não Normalizada

É a padronização da representação das fontes de dados no formato de tabela aninhada. Neste caso a tabela deve conter valores atômicos e grupos de repetição.

Exemplo:

CódProj	Tipo	Descr	Emp					
			CodEmp	Nome	Cat	Sal	DataIni	TempAl
LSC001	Novo Desenv.	Sistema de Estoque	2146	João	A1	4	1/11/91	24
			3145	Sílvio	A2	4	2/10/91	24
			6126	José	B1	9	3/10/92	18
			1214	Carlos	A2	4	4/10/92	18
			8191	Mário	A1	4	1/11/92	12
PAG02	Manutenção	Sistema de RH	8191	Mário	A1	4	1/05/93	12
			4112	João	A2	4	4/01/91	24
			6126	José	B1	9	1/11/92	12

Projetos (codProj, tipo, descr,
(codEmp, nome, cat, sal, dataIni, tempoAloc))

Figura 65 - Tabela não normalizada

8.2.3 Normalização

É a decomposição sistemática da tabela não normalizada em várias tabelas relacionais. É um processo baseado na aplicação de regras (formas normais).

O objetivo é a eliminação de redundância no armazenamento e organização dos dados em entidades lógicas.

8.2.4 Integração de Esquemas Relacionais/Projeto Lógico

É a obtenção do esquema relacional unificado para todas as fontes de dados normalizadas.

O objetivo é a integração de tabelas que mantém as mesmas entidades e relacionamentos com eliminação de tabelas redundantes.

8.2.5 Engenharia Reversa/Projeto físico ou implementação

Esta fase depende do modelo no qual se está trabalhando. Caso o modelo seja o de um banco de dados implementado será empregada a técnica de engenharia reversa.

Caso seja somente de dados organizados, baseado no projeto lógico gerar o projeto físico. Esta etapa pode ser realizada com o apoio de ferramentas case.

8.3 COMPARATIVO BÁSICO ENTRE AS ESTRATÉGIAS

***Top-Down*:** Gera esquemas de banco de dados baseados nos requisitos da organização obtidos através de contatos com os usuários.

***Bottom-Up*:** Gera esquemas de banco de dados baseados nas fontes de dados da organização.

Uma completa a outra.

LINGUAGEM DE INTERROGAÇÃO

Até então foram estudados conceitos relacionados a projeto de banco de dados. Este capítulo será destinado aos conceitos da álgebra e cálculo relacional aplicados como linguagem de interrogação a banco de dados, o que será muito útil na realização de consultas em uma próxima etapa da disciplina de Banco de Dados.

Aspecto importante a ressaltar sobre a álgebra relacional é que ela é uma área da Matemática que inspirou o modelo relacional.

9.1 LINGUAGENS DE CONSULTAS FORMAIS

Uma linguagem de consulta (*Query Language*) é uma linguagem com a qual o usuário pode requisitar ao Sistema de Gerência de Banco de Dados (SGBD) informações armazenadas no Banco de Dados (BD).

As linguagens de consulta são classificadas em dois tipos:

a) **Procedurais:** O usuário descreve o algoritmo de acesso aos dados através de uma seqüência de instruções (COMO)

b) **Não procedurais:** O usuário descreve a informação que deseja obter sem descrever como obtê-la (O QUÊ)

As linguagens de consulta e atualização comerciais para sistemas relacionais baseiam-se na Álgebra Relacional (procedural) e no Cálculo Relacional (não procedural).

As operações da álgebra e do cálculo exprimem o conjunto de consultas e manipulações possíveis sobre uma base de dados relacional qualquer.

A álgebra apresenta o conjunto mínimo de **operadores relacionais** que podem ser combinados para **extrair** da base de dados, praticamente, todas as **informações** ali armazenadas (dados e seus relacionamentos) ♉.

O cálculo estende (e completa) a potencialidade da álgebra relacional com a introdução dos quantificadores universal (∀) e existencial (∃).

9.2 CONCEITOS

- Relação: representada por uma tabela de duas dimensões (linhas e colunas);

- Tupla: corresponde a uma linha da relação;

- Atributo: corresponde às colunas da relação;

- Chave primária: conjunto de atributos que identificam univocamente cada tupla da relação;

- Chave estrangeira: atributo de uma relação que é chave primária de outra relação.

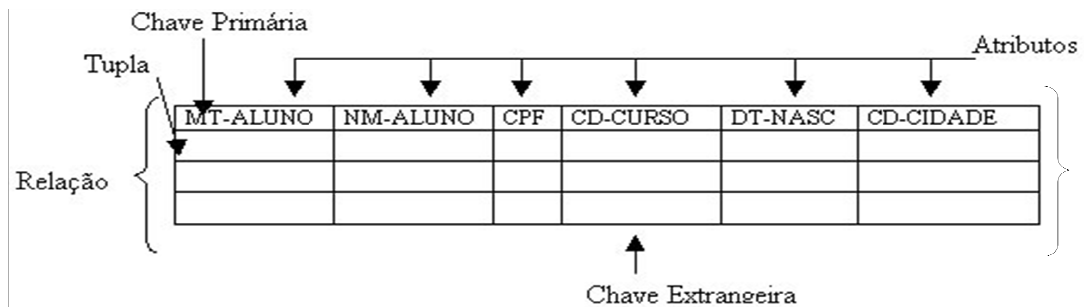



Figura 66 - Tabela não normalizada

9.2.1. Álgebra Relacional

A Álgebra Relacional é um conjunto de operações sobre modelos relacionais de dados. Podem ser agrupadas em duas categorias:

- Operadores Tradicionais:

· União (union): $\cup$

· Interseção: $\cap$

· Diferença (set-difference): -

· Produto Cartesiano (cartesian product): x

- Operadores Relacionais:

· Restrição/Seleção (select): σ

· Projeção (project): π

· Junção Theta: $\Xi\theta$

. Junção Natural: Ξ

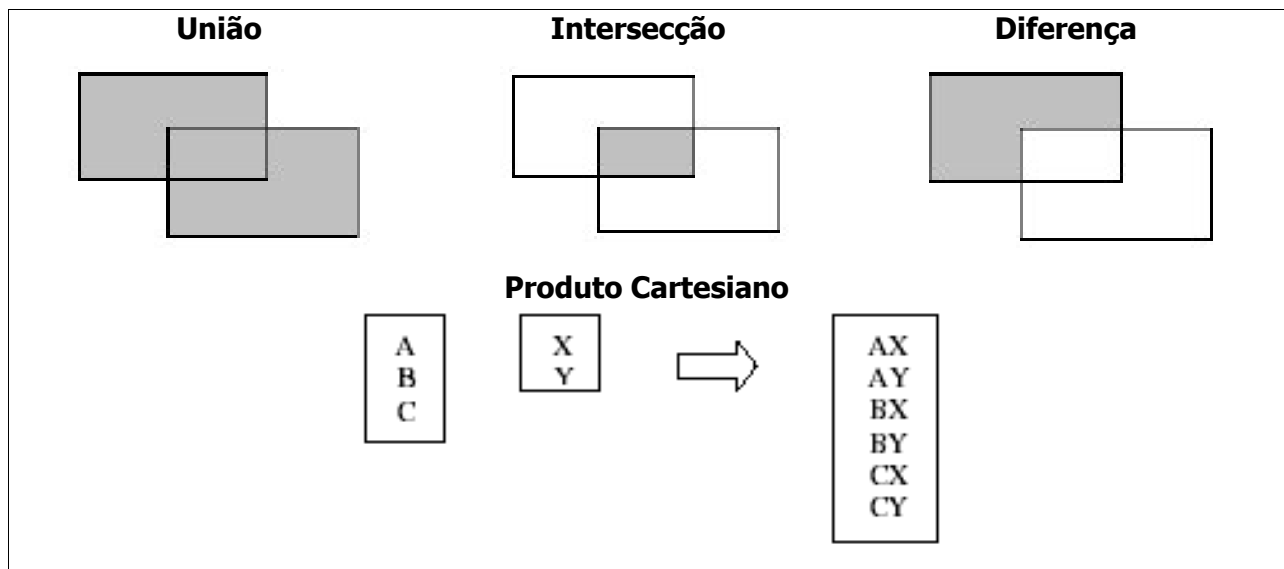
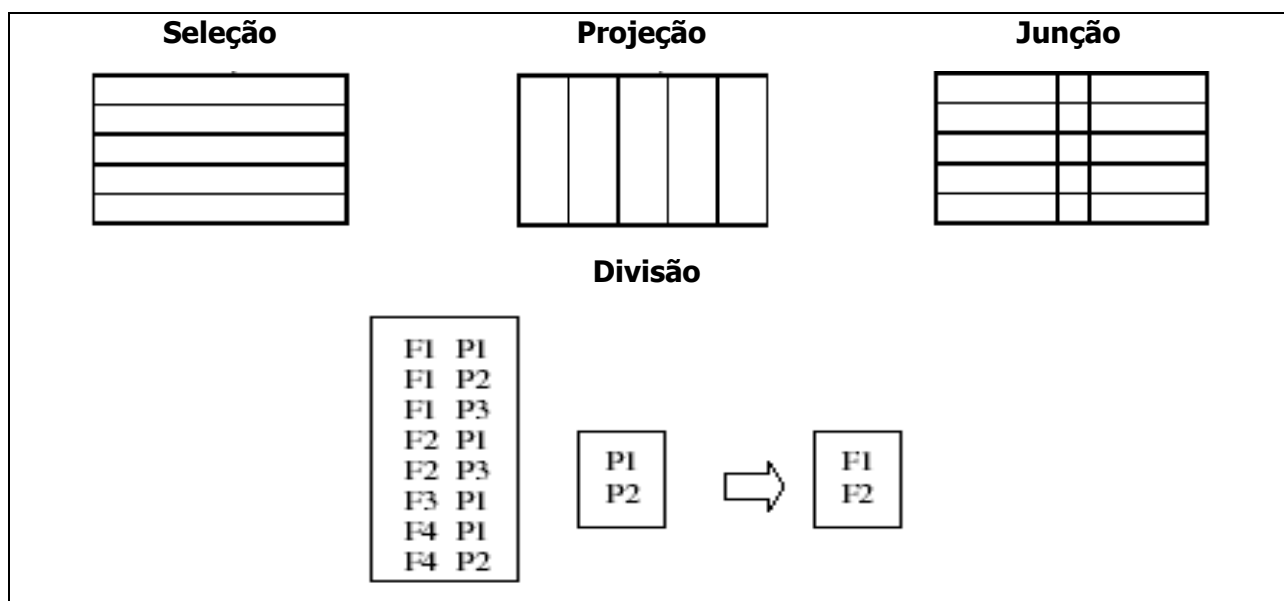
· Divisão: $\div$

A Seleção e Projeção são operações UNÁRIAS.

As outras três operações (Produto Cartesiano, União e Diferença) operam, cada uma, sobre um par de relações.

As operações da Álgebra Relacional sempre operam sobre relações e devolvem como resultado uma relação.

9.2.2. Representação Gráfica

9.2.2.1. OPERADORES TRADICIONAIS

9.2.2.2. OPERADORES RELACIONAIS

9.2.2.3. SIMBOLOGIA

- União $R \cup S$
- Intersecção $R \cap S$
- Diferença $R - S$
- Prod. Cartesiano $R \times S$
- Seleção $\sigma_F (R)$
- Projeção $\Pi_{i_1, i_2, \ldots, i_m}(R)$
- Junção $R \bowtie S$
- Divisão R / S

9.2.2.4. EXEMPLIFICAÇÃO

Para entender os conceitos da Álgebra Relacional observe as relações abaixo:

Fornecedor

#F	Denominação	Condição	Cidade
F1	Pedro	20	Joinville
F2	João	10	Florianópolis
F3	Marcos	30	Florianópolis
F4	Carlos	20	Joinville
F5	Ademir	30	Laguna

Produto

#P	Descrição	Valor
P1	Parafuso	20
P2	Arroela	10
P3	Porca	30
P4	Chave	20
P5	Fechadura	30

#F	#P	Quantidade
F1	P1	300
F1	P2	200
F1	P3	400
F1	P4	200
F1	P5	100
F1	P6	100
F2	P1	300
F2	P2	400
F3	P2	200
F4	P2	200
F4	P4	300
F4	P5	400

a. União - R $\cup$ S

A união de duas relações A e B é o conjunto de todas as tuplas pertencentes a relação A ou pertencentes a relação B.

Exemplo:

A = conjunto de tuplas de forncedores de “Joinville”

B = conjunto de tuplas de fornecedores que fornecem “P1”

C = A união B

#F	Nome	Condição	Cidade
F1	Pedro	20	Joinville
F2	João	10	Florianópolis
F4	Carlos	20	Joinville

b. Intersecção - R ∩ S

A intersecção de duas relações A e B é o conjunto de todas as tuplas pertencentes a relação A e pertencentes a relação B.

Exemplo:

A = conjunto de tuplas de fornecedores de “Joinville”

B = conjunto de tuplas de fornecedores que fornecem “P1”

C = A intersecção B

#F	Nome	Condição	Cidade
F1	Pedro	20	Joinville

c. Diferença - R – S

A diferença de duas relações A e B é o conjunto de todas as tuplas pertencentes a relação A e não pertencentes a relação B.

Exemplo:

A = conjunto de tuplas de fornecedores de “Joinville”

B = conjunto de tuplas de fornecedores que fornecem “P1”

C = A diferença B

#F	Nome	Condição	Cidade
F4	Carlos	20	Joinville

d. Produto Cartesiano - R x S

O produto cartesiano de duas relações A e B é o conjunto de todas as tuplas t originadas da concatenação das tuplas *a* pertencentes a A e das tuplas *b* pertencentes a B.

Exemplo:

A = conjunto de todos os códigos de fornecedores de “Joinville”

B = conjunto de todos os códigos de produtos de cor “azul”

C = A produto B

#F	#P
F1	P3
F1	P5
F4	P3
F4	P5

e. Seleção - σ_F (R)

É a operação usada para construir um subconjunto horizontal de uma relação, cujas tuplas satisfaçam uma determinada condição.

Exemplo:

C = SELEÇÃO(fornecedor, (cidade = “Joinville”))

$\sigma_{Cidade=Joinville}$ (Fornecedor)

#F	Nome	Condição	Cidade
F1	Pedro	20	Joinville
F4	Carlos	20	Joinville

f. Projeção - $\Pi_{i1, i2, ..., im}$(R)

É a operação usada para construir um subconjunto vertical de uma relação, cujas tuplas satisfaçam uma determinada condição.

Exemplo:

C = PROJEÇÃO(fornecedor, (#F, denominação, cidade))

$\Pi_{\#F,denominação, cidade,}$ (Fornecedor)

#F	Denominação	Cidade
F1	Pedro	Joinville
F2	João	Florianópolis
F3	Marcos	Florianópolis
F4	Carlos	Joinville

g. Junção - R ⋈ S

De duas relações R1 e R2, que possuem um atributo em comum D, é o subconjunto do produto cartesiano das duas relações, cujos valores dos elementos do atributo comum sejam iguais nas duas relações.

Na relação resultante elimina-se a repetição da coluna D.

Exemplo:

C = JUNÇÃO(fornecedor, pedido(#F))

Fornecedor ⋈ Pedido

#F	Denominação	Condição	Cidade
F1	Pedro	20	Joinville
F1	Pedro	20	Joinville
F1	Pedro	20	Joinville
F1	Pedro	20	Joinville
F1	Pedro	20	Joinville
F1	Pedro	20	Joinville
F2	João	10	Florianópolis

h. Divisão - R / S

Seja A uma relação binária com atributos x e y e B uma relação unária com atributo z, com y e z definidos sobre o mesmo domínio. Definimos a operação divisão, como sendo o conjunto dos elementos x com os pares (x,y) pertencentes a A para todos os valores y pertencentes a B.

Exemplo:

C = DIVISÃO(A, B ((#P))

9.3 LINGUAGENS DE INTERROGAÇÃO

Álgebra Relacional pode ser usada como linguagem de interrogação à Banco de Dados.

Exemplos:

- Quais os nomes dos professores do curso de Ciência da Computação?

Π_{nome} [$\sigma_{curso\ =\ '104'}$ (Professor)]

Professor

#prof	Nome	Curso
10	João	104
20	Pedro	102
30	Paulo	104
40	Marcos	101

- Quais os nomes e datas de nascimento dos alunos do curso 'SI' nascidos antes de 1983?

$\Pi_{nome,\ data_nasc}$ [$\sigma_{codcurso\ =\ '104'\ \wedge\ data_nasc\ <\ 1983\text{-}01\text{-}01}$ (Aluno)]

Aluno

#alu	Nome	Curso	Dt_nas
10	João	104	01/02/81
20	Pedro	102	05/10/83
30	Paulo	104	07/06/86
40	Marcos	101	07/02/89

Considere agora as relações abaixo: **Clientes, Depósitos e Empréstimos**, respectivamente.

c-nome	*c-endereço*	*c-cidade*
Jones	Main	Harrison
Smith	North	Rye
Hayes	Main	Harrison
Curry	North	Rye
Lindsay	Park	Pittsfield
Turner	Putnam	Stamford
Williams	Nassau	Princeton
Adams	Spring	Pittsfield
Johnson	Alma	Palo Alto
Glenn	Sand Hill	Woodside
Brooks	Senator	Brooklyn
Green	Walnut	Stamford

d-agência	*d-conta*	*d-nome*	*d-saldo*
Downtow	101	Johnson	500
Mianus	215	Smith	700
Perryridg	102	Hayes	400
Round	305	Turner	350
Perryridg	201	Williams	900
Redwood	222	Lindsay	700
Brighton	217	Green	750

e-agência	*e-código*	*e-nome*	*e-valor*
Downtown	17	Jones	1000
Redwood	23	Smith	2000
Perryridge	15	Hayes	1500
Downtown	14	Jackson	1500
Mianus	93	Curry	500
Round Hill	11	Turner	900
Pownal	29	Williams	1200
North	16	Adams	1300
Downtown	18	Johnson	2000
Perryridge	25	Glenn	2500
Brighton	10	Brooks	2200

- Selecione as tuplas da relação EMPRÉSTIMOS para qual o nome da agência é "Perryridge":

R = σe-agência='Perryridge' (EMPRÉSTIMOS)

Em geral, os predicados permitem expressar comparações do tipo (<, £, >, ³, = e ¹).

Além disso, pode-se relacional com operadores lógicos (*and, or, not*).

- Selecione tuplas da relação EMPRÉSTIMOS para as quais o valor do empréstimo é maior que 1200:

R = **σe-valor>1200 (EMPRÉSTIMOS)**

- Selecione as tuplas da relação EMPRÉSTIMOS para as quais o nome da agência é "Perryridge e o valor do empréstimo excede 1200:

R = **σe-valor > 1200 and e-agencia** = **'Perryridge'(EMPRÉSTIMOS)**